TAMPEREEN TEKNILLINEN YLIOPISTO
TAMPERE UNIVERSITY OF TECHNOLOGY

Jani Vilppo

IMPLEMENTING A CONTINUUM DAMAGE MATERIAL MODEL INTO ELMER FEM-SOFTWARE

Master of Science Thesis

Examiner: Professor Reijo Kouhia

The examiner and topic of the thesis
were approved on 30 May 2018

ABSTRACT

JANI VILPPO: Implementing a Continuum Damage Material Model into Elmer FEM-software
Tampere University of Technology
Master of Science Thesis, 48 pages, 25 Appendix pages
September 2018
Master's Degree Programme in Mechanical Engineering
Major: Applied Mechanics and Thermal Sciences
Examiner: Professor Reijo Kouhia

Keywords: concrete, constitutive model, CDM, damage, Ottosen failure criterion, Elmer, UMAT, FEM

Quasi-brittle materials like concrete are widely used in the construction industry. The failure of these materials is mainly due to macro- and micro-cracking. Thus the nonlinear behaviour of these materials can be simulated well using Continuum Damage Mechanics approach. CDM utilizes damage state variables which indicate the damaging of the material due to cracking. In order to simulate complex loading scenarios, the continuum damage material model must be solved numerically using Finite Element Method, for instance.

In this thesis an existing continuum damage material model is implemented into an open-source Elmer FEM-software. The chosen material model utilizes second order damage tensor and Ottosen's 4-parameter damage criterion. A comprehensive review of the thermodynamical formulation is performed. Then a program that solves the resulting constitutive equations is developed and implemented into Elmer via its user-defined material model subroutine feature. Lastly some well-known test cases are simulated and results are compared to experimental data found in the literature.

The original material model managed to simulate well some simpler test cases but failed in more general test cases. However, the failure modes were predicted well. The Finite Element-implementation shows good preliminary results. The initial failure modes were predicted well even for complex loading cases. But as the model features rate-independent strain-softening combined with damage formation, the FE-solutions are strongly mesh-dependent and convergence problems occur due to strain localization. Some future improvement possibilities are discussed in order to refine the model into a useful engineering tool.

TIIVISTELMÄ

JANI VILPPO: Vaurioituvan kontinuumimateriaalimallin ohjelmointi Elmer elementtimenetelmäohjelmistoon
Tampereen teknillinen yliopisto
Diplomityö, 48 sivua, 25 liitesivua
Syyskuu 2018
Konetekniikan diplomi-insinöörin tutkinto-ohjelma
Pääaine: Sovellettu mekaniikka ja lämpötekniikka
Tarkastaja: professori Reijo Kouhia

Avainsanat: betoni, konstitutiivinen malli, CDM, vaurio, Ottosenin vauriokriteeri, Elmer, UMAT, FEM

Kvasi-haurita materiaaleja, kuten betonia, käytetään paljon rakennusteollisuudessa. Niiden vaurioituminen johtuu pääasiassa makro- ja mikrohalkeilusta. Siksi näiden materiaalien epälineaarista käyttäytymistä voidaan mallintaa käyttämällä jatkuvan vaurion malleja. Siinä materiaalin halkeilusta johtuvaa lujuuden menetystä kuvataan vauriomuuttujalla. Monimutkaisten rakenteiden simuloimiseksi materiaalin käyttäytyminen pitää ratkaista numeerisesti käyttämällä esimerkiksi elementtimenetelmää.

Tässä työssä eräs jo olemassa oleva vaurioituva kontinuumimateriaalimalli implementoidaan Elmeriin, joka on avoimen lähdekoodin elementtimenetelmäohjelmisto. Implementoitava materiaalimalli käyttää toisen asteen vauriotensoria ja Ottosenin 4-parametrista vaurioehtoa. Materiaalimallin termodynaamiset perusteet esitellään. Sen jälkeen käsitellään materiaalimallin konstitutiiviset yhtälöt ratkaisevaa aliohjelmaa ja implementoidaan se Elmer-ohjelmistoon hyödyntäen mahdollisuutta antaa Elmerille käyttäjän määrittelemä materiaalimalli. Lopuksi mallin antamia tuloksia verrataan kirjallisuudesta löytyviin testitapauksiin.

Työssä käytettävä konstitutiivinen malli toimii hyvin yksinkertaisimmissa testitapauksissa, mutta ongelmia esiintyy monimutkaisemmissa tapauksissa. Vauriomuodon se kuitenkin ennustaa hyvin. Elementtimenetelmän avulla saatiin hyviä alustavia tuloksia varsinkin alustaville vauriomuodoille. Mutta koska materiaalimalli perustuu kuormitusnopeudesta riippumattomaan pehmenemismalliin ja vauriomuodostukseen, elementtimenetelmällä saatavat ratkaisut ovat hyvin verkkoriippuvaisia. Tuloksista nähdään, että muodonmuutokset lokalisoituvat pienelle alueelle, ja malli lakkaa konvergoimasta. Joitakin kehitysehdotuksia on annettu työn loppupuolella.

PREFACE

At the time this thesis topic was offered to me, I was very excited. It allowed me to learn more about continuum mechanics, which itself was a very interesting topic. Moreover, I would gain experience on a free, open source FEM-software environment. Furthermore, the idea of using Fortran programming language sounded like a proper mechanical engineer's way of using modern-day punch card–less programming methods in practice.

The overwhelming excitement was overcome when the partial differentiations led to 4th order tensor products and the length of the dissipation potential derivatives with respect to thermodynamic forces did not quite fit any paper I had on hand. But with good and rather challenging help from the instructor, professor Reijo Kouhia, we eventually managed to reach the colorful FEM pictures.

I also want to acknowledge and apologize all the third party-victims of this project — social relationships, 2.0 16v turbo Sierra-project and my own personal free time. But in the end this hard work has been very rewarding and I have learned a lot about mechanics during this thesis.

In Tampere, Finland, on 18 September 2018

Jani Vilppo

CONTENTS

LIST OF MATHEMATICAL SYMBOLS

A	Material parameter, surface area
a_1, a_2	Material parameters
$\boldsymbol{B}$	Kinematic matrix
B	Material parameter, volume of arbitrary body B
∂B	Surface of arbitrary body B
$\boldsymbol{b}$	Body force vector
$\boldsymbol{C}, \boldsymbol{C}_{ATS}$	Secant and Algorithmic tangential elasticity tensor
C_e	Elastic complementary strain energy
$\boldsymbol{D}$	Damage tensor
D_{ij}	Damage tensor component ij
E	Elastic modulus
f	Damage surface, general continuum property
g	A rational function
H_0	Material parameter
$\boldsymbol{I}, \boldsymbol{II}$	Second and fourth order identity tensors
I_1	First invariant of the stress tensor
I_Σ	Indicator function
$\boldsymbol{J}_\mathrm{e}$	Geometric Jacobian matrix
J_2	Second invariant of the deviatoric stress tensor
$\tilde{J}_2$	Reformulated second invariant of the deviatoric stress tensor
J_3	Third invariant of the deviatoric stress tensor
$\tilde{J}_3$	Reformulated third invariant of the deviatoric stress tensor
$\boldsymbol{K}_\mathrm{T}$	Linearized tangential stiffness matrix
K	Kinetic energy, thermodynamic force
k_1, k_2	Material parameters
$\boldsymbol{L}, \boldsymbol{L}_{ATS}$	Secant and Algorithmic tangential compliance tensor
m	Order of polynomial function
$\boldsymbol{N}$	Antsatz shape function vector
N	Antsatz shape function
$\boldsymbol{n}$	Unit normal vector
P	Force
P^{mech}	Mechanical power
$\boldsymbol{Q}$	Proper orthogonal second order tensor
R	Heat transfer rate
$\boldsymbol{q}$	Heat flux vector, nodal displacement vector
r	Distributed heat source
$\boldsymbol{S}$	Set of state variables
S	Entropy
$\boldsymbol{s}$	Deviatoric stress tensor
s	Entropy per unit mass

T	Absolute temperature
T^{RHS}	Absolute temperature of a reversible heat source
$\boldsymbol{t}$	Surface traction force vector
$\boldsymbol{u}$	Displacement vector
$\boldsymbol{Y}$	Thermodynamic force
U	Internal energy
u	Internal energy per unit mass
V	Volume
$\boldsymbol{W}$	Set of thermodynamic forces
W_e	Elastic strain energy
W_p	Weight factor
$\boldsymbol{x}$, $\dot{\boldsymbol{x}}$, $\ddot{\boldsymbol{x}}$	Global coordinate vector, velocity vector and acceleration vector
$\boldsymbol{Z}$	Set of variables
α_1, α_2	Material parameters
γ	Power of dissipation
δ_{ij}	Kronecker delta
$\boldsymbol{\varepsilon}$	Infinitesimal strain tensor, engineering strain vector
ε_bc	Ultimate biaxial compression strain
ε_c	Ultimate uniaxial compression strain
ε_t	Ultimate uniaxial tension strain
$\Delta\boldsymbol{\varepsilon}$	Strain increment
θ	Lode angle
$\boldsymbol{\kappa}$	Set of hardening-softening variables
κ	Hardening-softening variable
κ_0	Material parameter
Λ	A term in the Ottosen's damage criterion
$\dot{\lambda}$	A multiplier
ν	Poisson's ratio
$\boldsymbol{\xi}$	Reference element coordinate vector
ξ_i	Reference element coordinate vector component i
ρ, ρ_0	Density and initial density
Σ	A convex set
$\boldsymbol{\sigma}$	Stress tensor or vector
$\partial\Delta\boldsymbol{\sigma}/\partial\Delta\boldsymbol{\varepsilon}$	Incremental material Jacobian matrix
σ_bc, σ_bc0	Ultimate and initial elastic biaxial compression strength
σ_c, σ_c0	Ultimate and initial elastic uniaxial compression strength
σ_e4, σ_m4	Ultimate strength values in compressive meridian
σ_{ij}	Stress tensor component ij
σ_t, σ_t0	Ultimate and initial elastic uniaxial tension strength
φ	Dissipation potential
ψ	Helmholtz free energy per unit mass
ψ^c	Complementary Helmholtz free energy per unit mass

1. INTRODUCTION

Concrete is an example of a very common quasi-brittle material. It is the most used construction material due to its good availability, simple production and possibility of casting it on the construction site. Therefore it has been under a lot of investigation over last decades. (Peck 2012)

Typical properties of quasi-brittle materials like concrete are relatively high compressive strength and low tensile strength. Typical ultimate compressive strengths of concrete are in the range of 10–60 MPa, while corresponding tensile strength is only about 10 % of that. This means that the behaviour of damaged material is *highly anisotropic* as it depends on the type of loading. It has been revealed that the failure is mainly due to co-operative action of an array of micro-cracks in compression and the growth of the most critical micro-crack in tension (Dragon et al. 2000). This causes failure modes to be very different as illustrated in figure 1.1.

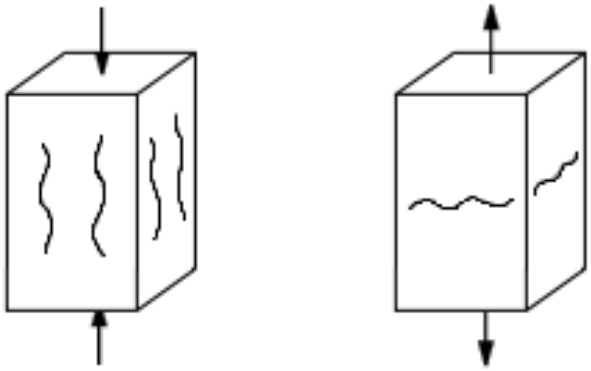


Figure 1.1. *Typical crack patterns in (a) uniaxial compression and (b) uniaxial tension.*

Typical design approach is to neglect concrete's tensile strength entirely and use steel reinforcement bars to cope with the tensile loads. In applications where concrete must take tensile loads into account, concrete anchors for instance, many different failure criterion have been developed. Examples of such failure criterion are the Drucker-Prager criterion (Drucker and Prager 1952), Mohr-Coulomb criterion with tension cut-off (Coulomb 1776; Mohr 1900) and Ottosen's criterion (Ottosen 1977). The design criterion is then that the concrete must stay in the elastic region determined by the failure criterion. But in order to simulate the ultimate limit state of concrete structures, the nonlinear behaviour of the concrete after initial yielding must be studied.

The conventional method for modelling the nonlinear behaviour of concrete beyond the elastic limit is to model it as elasto-plastic continuum. However, the nonlinearities in

quasi-brittle materials are mostly due to micro-cracking and plasticity has only small role in the process. The cracking is modelled by introducing a *damage state variable* into the model. Approach is called *Continuum Damage Mechanics* (CDM). Scalar damage variable has been first introduced by Kachanov (1958). However, in order to properly describe the complex failure process of concrete, vector (Kachanov 1974), second order tensor (Murakami 1988) and even higher order damage tensors have been utilized.

The thermodynamical framework of nonlinear behaviour due to damaging and strain-hardening–softening is reviewed in chapter 2. The specific concrete material model based on the second order damage tensor and Ottosen's failure criterion is also revisited in that chapter. The model is based on the works done by Yaghoubi et al. (2014) and Hartikainen et al. (2018).

In order to simulate the behaviour of complex structures, the specific model is implemented into an open-source *Finite Element* (FE) -software Elmer. Elmer software package and nonlinear *Finite Element Method* (FEM) is reviewed in chapter 3. Special emphasis is given on user-defined nonlinear material model solve procedure. The implementation of the specific material model into Elmer FE-software is explained in chapter 4.

The FEM-implementation of the material model is verified by duplicating the numerical results by Hartikainen et al. in chapter 5. Then some well-known test cases are simulated and compared to the experimental findings found in the literature. Finally, general overview of the results along with future improvement suggestions are given in chapter 6.

The main purpose of this work is to test the behaviour of the specific concrete material model in FE-applications. This thesis lays groundwork for further development of the model. Furthermore, a continuum damage material model based entirely on open-source FE-software is produced.

2. CONSTITUTIVE MODELLING

2.1 Nonlinear Material Models

Linear relation between stress and strain is called Hooke's law. In relatively small deformations most solid materials can be approximated with linear stress-strain response. Some materials show linear behaviour till moderately high loads.

Nonlinear elasticity becomes essential when greater loads are applied. Elastic behaviour means that all mechanical work done to the system is stored as internal strain energy. Therefore the process is *thermodynamically reversible*, and all work done to the system can be extracted back from the system. Even so, the elastic stress-strain relation can be nonlinear and thus needs nonlinear solution methods. These reversible, nonlinear materials are called hyper-elastic materials. Other reversible nonlinear type of material behaviour is visco-elasticity, in which the deformations are time-dependent but reversible. (Ottosen and Ristinmaa 2005)

In real materials, always some kind of *dissipation* occur during loading and thus such processes are *thermodynamically irreversible*. In the scope of constitutive modelling, presence of dissipation means that some portion of the work done to the system is not stored as internal energy. Work can be transformed into heat, plastic deformations, crack formation or other forms. These processes are determined ultimately by the micro-structure of materials. However, in continuum mechanics the material behaviour is described by approximating micro-level physics in macro-scale. Thus we have to develop constitutive models that represent the micro-level behaviour well enough for the application. Because different materials have different micro-structures the constitutive relations differ from material to material.

As the failure of quasi-brittle materials is mainly due to the crack formation, obvious choice is to utilize damage state variable in the model. The order of damage tensor depends on the desired accuracy of the model. Higher order tensorial damage representations are more accurate than lower order tensors. They are also more complex and computationally intensive. In addition to cracking, also the elastic domain of concrete changes during irreversible loading. Thus it is justifiable to take *strain-hardening–softening* into account. (Murakami 2012)

At lower loads the undamaged concrete behaves like isotropic linear elastic solid. The limit between elastic and inelastic behaviour is described by the damage surface. Inside the damage surface material behaviour is linearly elastic, but in case that some damaging has already occurred, the behaviour can be anisotropic. The actual damage surface must be chosen such that it represents the nature of concrete.

2.2 General Principles of Constitutive Modelling

Next we present the basic theory of constitutive modelling. We do not focus on the specific model yet. However, the presented theory will be directed towards modelling damaging and strain-softening–hardening concrete materials. The aim is to explain the thermodynamical background of the specific material model that will be implemented into Elmer. The basic theory is adapted from the books of Malvern (1969), Ottosen and Ristinmaa (2005), Tadmor et al. (2012) and Murakami (2012).

2.2.1 Fundamental Principles of Constitutive Modelling

Continuum systems are governed by a set of physical laws which are the *conservation of mass* (2.1), *balance of linear momentum* (2.2) and *conservation of energy* (2.3). Furthermore, *balance of angular momentum* (2.4) and *Clausius–Duhem inequality* (2.5) cause extra restrictions to the system. (Tadmor et al. 2012, p. 180)

$$\dot{\rho} + \rho(\operatorname{div}\dot{\boldsymbol{x}}) = 0 \tag{2.1}$$

$$\operatorname{div}\boldsymbol{\sigma} + \rho\boldsymbol{b} = \rho\ddot{\boldsymbol{x}} \tag{2.2}$$

$$\boldsymbol{\sigma} : \dot{\boldsymbol{\varepsilon}} + \rho r - \operatorname{div}\boldsymbol{q} = \rho\dot{u} \tag{2.3}$$

$$\boldsymbol{\sigma}^T = \boldsymbol{\sigma} \tag{2.4}$$

$$\dot{s} \geq \frac{r}{T} - \frac{1}{\rho}\operatorname{div}\frac{\boldsymbol{q}}{T} \tag{2.5}$$

Equations (2.1) to (2.5) are given here as information only and thus the meaning of each symbol is given in the *List of Mathematical Symbols*.

These physical laws produce a system of equations that contains more unknowns than equations. Therefore *constitutive relations* are needed to close the system. Relations cannot be arbitrary. Tadmor et al. describe 5 fundamental principles for constitutive relations. Constitutive relations can be developed entirely based on these principles or the model can be checked afterwards to make sure that all constraints are fulfilled.

I Principle of determinism

This is a fundamental philosophical principle that states that past events determine the present. In the sense of constitutive modelling this means that the state of any physical variable can be determined if the present and past values of other variables are known. Especially in history-dependent material models the current state depends on the load history.

II Principle of local action

This principle states that the material response depends only on the conditions very close to the material point. This allows us to define material response point-wise and then integrate

over all material points to capture the behaviour of a finite body or a finite element, like described in section 3.2. However, also nonlocal continuum theories exist but they are not discussed in this thesis.

III Second law restrictions

The second law of thermodynamics states that the entropy of any isolated system either remains constant or increases. This principle must hold for any constitutive relation. This principle takes the form of the Clausius-Duhem inequality for thermomechanical continuum systems. The *Coleman-Noll procedure* utilized in section 2.2.4 is based on the second law of thermodynamics and therefore produces thermodynamically admissible relations by definition.

IV Principle of material frame-indifference (objectivity)

Constitutive relations must rely only on objective physical variable fields. Objective physical variable is physically independent of the frame of reference. For example the relative position between two points is the same in all frames of reference, but an absolute position is not.

V Material symmetry

If we are modelling a material that has any kind of symmetries, the constitutive relations must take them into account. For example homogeneous isotropic material has to act similarly regardless of the initial direction of the material before loads are applied.

2.2.2 Damage Surface

Damage surface is a surface that defines a domain of reversible processes. Usually in structural mechanics it defines the elastic domain. Material behaves elastically if its state stays inside the damage surface during the whole deformation process. In other words the mechanical behaviour is thermodynamically reversible. Reversible and irreversible behaviour is described in greater detail in the following sections.

The general form of damage surface f is

$$f(\boldsymbol{S}) = 0 \tag{2.6}$$

and the elastic domain is then

$$f(\boldsymbol{S}) \leq 0. \tag{2.7}$$

The damage surface depends on a set of state variables $\boldsymbol{S}$. They may include stress $\boldsymbol{\sigma}$, strain $\boldsymbol{\varepsilon}$, temperature T, damage $\boldsymbol{D}$, hardening-softening variables $\boldsymbol{\kappa}$, etc. There can also be a dependance on time or time derivatives. Usually there must be also some material

parameters that need to be calibrated with test data to make sure the damaging is consistent with real materials.

Elastic domain can change during irreversible loading. For example hardening-softening variables can control material's isotropic or kinematic hardening-softening. Isotropic hardening-softening changes the size of elastic domain, while kinematic hardening-softening changes its position in the state variable space. Real materials usually exhibit some isotropic and some kinematic hardening. The original, unchanged damage surface is called the *initial damage surface*. (Ottosen and Ristinmaa 2005, section 9.1)

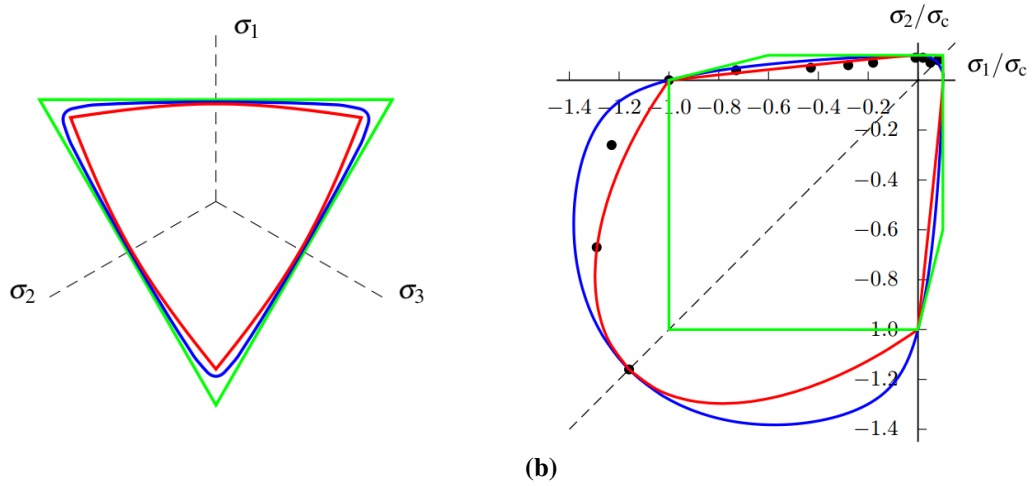
Damage surface can be used as a design criterion. A common design criteria for structures is that they must stay in the elastic region. Then the initial damage surface is the failure criterion. Some failure criterion for concrete include:

- 2-parameter Drucker-Prager criterion (Drucker and Prager 1952)
- 3-parameter Mohr-Coulomb criterion with tension cut-off (Coulomb 1776; Mohr 1900)
- 3-parameter so-called Barcelona model (Lubliner et al. 1989)
- 4-parameter Ottosen's criterion (Ottosen 1977).

The characteristic shapes of some failure surfaces are illustrated in figure 2.1.

Ottosen's 4-parameter failure criterion is used in the specific material model that is developed in later chapters. It takes the form

$$f(I_1, J_2, \cos 3\theta) = \frac{A}{\sigma_c} J_2 + \Lambda \sqrt{J_2} + BI_1 - \sigma_c = 0. \tag{2.8}$$



(a) **(b)**

Figure 2.1. *Characteristic shapes of three different failure criterion (a) in the π-plane and (b) in plane stress state. Green line represents Mohr-Coulomb criterion with tension cut-off, red is the Barcelona criterion and blue is Ottosen's criterion. Black dots indicate experimental results by Kupfer et al. (1969). Figures by Hartikainen et al. (2018)*

It depends on the first invariant of the stress tensor $I_1 = \mathrm{tr}\,\boldsymbol{\sigma}$, the second invariant $J_2 = \frac{1}{2}\mathrm{tr}(\boldsymbol{s}^2)$ of the deviatoric stress tensor $\boldsymbol{s} = \boldsymbol{\sigma} - \frac{1}{3}I_1\boldsymbol{I}$ and the Lode angle $\cos 3\theta$ that is defined in equation (2.10). Term $\Lambda = \Lambda(\cos 3\theta)$ is defined as

$$\Lambda = \begin{cases} k_1 \cos[\frac{1}{3}\arccos(k_2 \cos 3\theta)] & \text{if} \quad \cos 3\theta \geq 0 \\ k_1 \cos[\frac{1}{3}\pi - \frac{1}{3}\arccos(-k_2 \cos 3\theta)] & \text{if} \quad \cos 3\theta \leq 0 \end{cases} . \tag{2.9}$$

Lode angle is the angle in the deviatoric plane defined as

$$\cos 3\theta = \frac{3\sqrt{3}}{2}\frac{J_3}{J_2^{3/2}}, \tag{2.10}$$

where $J_3 = \frac{1}{3}\mathrm{tr}(\boldsymbol{s}^3)$ is the third invariant of deviatoric stress tensor.

The criterion contains 4 parameters and the uniaxial compressive stress σ_c that need to be calibrated with the experiments. Material parameters A and B are dimensionless. The size factor k_1 and the shape factor k_2 are also dimensionless material parameters. The failure surface is convex and smooth if the following conditions are satisfied:

$$A \geq 0, \qquad B \geq 0, \qquad k_1 \geq 0, \qquad 0 \leq k_2 \leq 1. \tag{2.11}$$

4 different failure stress states from experimental tests are needed to calibrate material parameters. Ottosen suggests following tests:

1. Uniaxial compression test σ_c
2. Biaxial compression test σ_{bc}
3. Uniaxial tension test σ_t
4. A failure state $(\sigma_{m4}, \sigma_{e4})$ or least square fit of multiple states along the compressive meridian.

Material parameters can then be solved from equations (2.8) – (2.10).

2.2.3 First and Second Laws of Thermodynamics

The *first law of thermodynamics* states that the total energy of a thermodynamic system and its surroundings is conserved (Tadmor et al. 2012, chapter 5). Thus the heat supplied to any finite system and mechanical work done to the system is stored as internal energy and kinetic energy in the system. In rate-form it can be expressed as

$$\dot{K} + \dot{U} = P^{mech} + R, \tag{2.12}$$

where $\dot{K}$ and $\dot{U}$ are the linear time derivatives of kinetic and internal energy of the system, respectively. On the right side P^{mech} is the power of external mechanical work done to the system and R is the rate of heat transfer. For a macroscopic body with the volume B and

outer surface ∂B, the kinetic energy, internal energy, mechanical power and heat power are defined as

$$K = \int_B \frac{1}{2}\rho \|\dot{\boldsymbol{x}}\|^2 \, dV, \tag{2.13}$$

$$U = \int_B \rho u \, dV, \tag{2.14}$$

$$P^{mech} = \int_B \rho \boldsymbol{b} \cdot \dot{\boldsymbol{x}} \, dV + \int_{\partial B} \boldsymbol{t} \cdot \dot{\boldsymbol{x}} \, dA, \tag{2.15}$$

$$R = \int_B \rho r \, dV - \int_{\partial B} \boldsymbol{q} \cdot \boldsymbol{n} \, dA, \tag{2.16}$$

where ρ is density, $\dot{\boldsymbol{x}}$ velocity, u internal energy per unit mass (or specific internal energy), $\boldsymbol{b}$ body forces acting on body B, $\boldsymbol{t}$ surface forces (or tractions) acting on the surface of body ∂B, r distributed heat source, $\boldsymbol{q}$ heat flux vector and $\boldsymbol{n}$ unit surface normal pointing outwards from surface ∂B.

We can combine above equations, apply *Reynolds transport theorem*, the *divergence theorem* and assume small deformations to form the *local form of the first law of termodynamics*

$$\rho r - \operatorname{div} \boldsymbol{q} = \rho \dot{u} - \boldsymbol{\sigma} : \dot{\boldsymbol{\varepsilon}}. \tag{2.17}$$

This equation is often referred to as the *energy equation*.

The *second law of thermodynamics* states that the entropy S of any isolated system either remains constant or increases. In mathematical form it can be expressed as

$$\Delta S \geq 0. \tag{2.18}$$

Alternatively the second law of thermodynamics can be formulated as

$$dS \geq \frac{dQ}{T^{RHS}}, \tag{2.19}$$

which is called the *Clausius–Planck inequality*. Differential dS is the entropy change of a closed system, dQ is the amount of heat supplied to the system by a reversible heat source and T^{RHS} is the temperature of the reversible heat source. We can then define expressions for the internal and external entropy generation as

$$dS^{ext} \equiv \frac{dQ}{T^{RHS}}, \tag{2.20}$$

$$dS^{int} \equiv dS - dS^{ext}. \tag{2.21}$$

By combining expressions (2.19) – (2.21) and differentiating with respect to time we end up with the rate form

$$\dot{S} \geq \dot{S}^{ext} = \frac{\dot{Q}}{T^{RHS}}. \tag{2.22}$$

The heat supply rate $\dot{Q}$ is the same as in equation (2.12). Entropy of the whole finite body is obtained by integrating the specific entropy s over the whole body:

$$S = \int_B \rho s \, dV. \tag{2.23}$$

Again, by combining above equations, applying Reynolds transport theorem, divergence theorem and realising that these equations must hold for any arbitrary body B, we get the *local form of the second law of termodynamics*

$$\dot{s} \geq \dot{s}^{ext} = \frac{r}{T} - \frac{1}{\rho} \operatorname{div} \frac{\boldsymbol{q}}{T}, \tag{2.24}$$

where $\dot{s}$-terms represent the *specific entropy generation rates*. In continuum thermodynamics theory it is assumed that the boundary points of any infinite body are at the same temperature as the possible reversible heat source. Thus T^{RHS} is simply the temperature T of the material point. (Tadmor et al. 2012, p. 175)

The equation (2.24) is called Clasius-Duhem inequality. The specific internal entropy generation rate $\dot{s}^{int}$ can be derived from definition (2.21) as

$$\dot{s}^{int} \equiv \dot{s} - \dot{s}^{ext}. \tag{2.25}$$

Because $\dot{s} \geq \dot{s}^{ext}$, it can be seen that specific internal entropy generation rate is always greater or equal to zero.

2.2.4 Thermodynamic Formulation of Constitutive Equations

Constitutive relations are developed using thermodynamically consistent method called the *Coleman-Noll procedure*, first presented by Coleman and Noll (1963). As the derivation is based on the first and second laws of thermodynamics, the resulting constitutive equations are thermodynamically admissible by definition.

Clasius-Duhem inequality (2.24) can be rewritten as

$$\dot{s}^{int} = \dot{s} - \dot{s}^{ext} = \dot{s} - \frac{r}{T} - \frac{1}{\rho} \operatorname{div} \frac{\boldsymbol{q}}{T} \geq 0. \tag{2.26}$$

Expanding the divergence term yields

$$\rho T \dot{s}^{int} = \rho T \dot{s} - [\rho r - \operatorname{div} \boldsymbol{q}] - \frac{1}{T} \boldsymbol{q} \cdot \operatorname{grad} T \geq 0. \tag{2.27}$$

The expression in square brackets is the same as left side of the energy equation (2.17). Substitution gives the *dissipation inequality*

$$\gamma = \rho T \dot{s} - \rho \dot{u} + \boldsymbol{\sigma} : \dot{\boldsymbol{\varepsilon}} - \frac{1}{T} \boldsymbol{q} \cdot \operatorname{grad} T \geq 0, \tag{2.28}$$

where the *power of dissipation* γ is defined as

$$\gamma \equiv \rho T \dot{s}^{int}. \tag{2.29}$$

Power of dissipation is zero for reversible processes and positive for irreversible processes. Because the dissipation inequality is derived directly from the laws of thermodynamics, any constitutive relation that satisfies the equation (2.28) is thermodynamically valid. (Coleman and Noll 1963)

Equation (2.28) is expressed with state functions u and s. If we define a new state function called the *specific Helmholtz free energy* ψ as

$$\psi = u - sT, \tag{2.30}$$

the dissipation inequality becomes

$$\gamma = -\rho(\dot{\psi} + s\dot{T}) + \boldsymbol{\sigma} : \dot{\boldsymbol{\varepsilon}} - \frac{1}{T}\boldsymbol{q} \cdot \operatorname{grad} T \geq 0. \tag{2.31}$$

Helmholtz free energy is the amount of internal energy that is available for doing useful work at constant temperature. The independent state variables in Helmholtz free energy are temperature T and kinematic variable, which in our case is the strain tensor $\boldsymbol{\varepsilon}$ (Malvern 1969, pp. 262-263). In isothermal case the Helmholtz free energy reduces to the *elastic strain energy* W_e (Ottosen and Ristinmaa 2005, section 4.1). It then measures the amount of reversible elastic energy stored in the system.

Alternatively we can use the complementary part of Helmholtz free energy. In isothermal case the *specific complementary Helmholtz free energy* ψ^{c} is obtained by applying the partial Legendre transformation to the specific Helmholtz free energy:

$$\psi^{\mathrm{c}} = \frac{1}{\rho}\boldsymbol{\sigma} : \boldsymbol{\varepsilon}^{\mathrm{e}} - \psi. \tag{2.32}$$

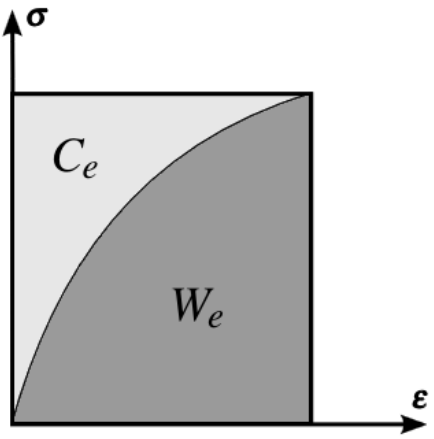


Figure 2.2. *Relation between strain energy W_e and complementary strain energy C_e. In isothermal case these are same as $\rho\psi$ and $\rho\psi^{\mathrm{c}}$, respectively.*

In complementary Helmholtz free energy the temperature T and stress tensor $\boldsymbol{\sigma}$ would act as independent state variables. But since we neglected thermal effects, the only independent state variable is stress. Furthermore, our material model does not deal with plastic deformations. Therefore elastic strains $\boldsymbol{\varepsilon}^{\mathrm{e}}$ are the same as total strains $\boldsymbol{\varepsilon}$. The relation between Helmholtz free energy and its complementary part is illustrated in figure 2.2. The complementary part represents the *elastic complementary strain energy* C_e in isothermal case.

Since we have assumed isothermal conditions, term $\dot{T}$ is zero in the dissipation inequality. If we further assume that there are no significant heat transfer, term $\boldsymbol{q}$ is also zero and the dissipation inequality (2.31) simplifies to form

$$\gamma = -\rho\,\dot{\psi} + \boldsymbol{\sigma} : \dot{\boldsymbol{\varepsilon}} \geq 0, \tag{2.33}$$

which is the *mechanical part of dissipation inequality* for isothermal case (Ottosen and Ristinmaa 2005, p. 561). It is defined in terms of the specific Helmholtz free energy.

By using the equation (2.32) we can express the mechanical part of dissipation inequality in terms of the specific complementary Helmholtz free energy as

$$\gamma = \rho\,\dot{\psi}^{\mathrm{c}} - \dot{\boldsymbol{\sigma}} : \boldsymbol{\varepsilon} \geq 0. \tag{2.34}$$

The exact form of the complementary Helmholtz free energy can be expressed in terms of external variables $\boldsymbol{\sigma}$ and T, as well as a set of internal variables (Murakami 2012, p. 71). These internal variables may include damage variable $\boldsymbol{D}$ and hardening-softening variable κ as in the specific model we develop in section 2.3. Keeping in mind that we have already declared isothermal conditions, the specific complementary Helmholtz free energy can now be expressed as

$$\psi^{\mathrm{c}} = \psi^{\mathrm{c}}(\boldsymbol{\sigma}, \boldsymbol{D}, \kappa). \tag{2.35}$$

Its material time derivative yields

$$\dot{\psi}^{\mathrm{c}} = \frac{\partial \psi^{\mathrm{c}}}{\partial \boldsymbol{\sigma}} : \dot{\boldsymbol{\sigma}} + \frac{\partial \psi^{\mathrm{c}}}{\partial \boldsymbol{D}} : \dot{\boldsymbol{D}} + \frac{\partial \psi^{\mathrm{c}}}{\partial \kappa}\dot{\kappa}. \tag{2.36}$$

Assuming constant density $\rho = \rho_0$ and substituting expression (2.36) to dissipation inequality (2.34) we obtain the form

$$\gamma = \left(\rho_0 \frac{\partial \psi^{\mathrm{c}}}{\partial \boldsymbol{\sigma}} - \boldsymbol{\varepsilon}\right) : \dot{\boldsymbol{\sigma}} + \rho_0 \frac{\partial \psi^{\mathrm{c}}}{\partial \boldsymbol{D}} : \dot{\boldsymbol{D}} + \rho_0 \frac{\partial \psi^{\mathrm{c}}}{\partial \kappa}\dot{\kappa} \geq 0. \tag{2.37}$$

In reversible processes the internal variables do not evolve. Thus in elastic region $\dot{\boldsymbol{D}} = \dot{\kappa} = 0$ and the equation (2.37) reduces to

$$\gamma = \left(\rho_0 \frac{\partial \psi^{\mathrm{c}}}{\partial \boldsymbol{\sigma}} - \boldsymbol{\varepsilon}\right) : \dot{\boldsymbol{\sigma}} \geq 0. \tag{2.38}$$

Equation (2.38) must hold for any arbitrary $\dot{\boldsymbol{\sigma}}$. Therefore the expression inside brackets ( ) must be zero, and we obtain the following constitutive relation:

$$\boldsymbol{\varepsilon} = \rho_0 \frac{\partial \psi^{\mathrm{c}}}{\partial \boldsymbol{\sigma}}. \tag{2.39}$$

We can then substitute equation (2.39) back into the dissipation inequality (2.37) to get the expression for *mechanical dissipation*:

$$\gamma = \rho_0 \frac{\partial \psi^{\mathrm{c}}}{\partial \boldsymbol{D}} : \dot{\boldsymbol{D}} + \rho_0 \frac{\partial \psi^{\mathrm{c}}}{\partial \kappa} \dot{\kappa} \geq 0. \tag{2.40}$$

If the system undergoes a reversible deformation, no dissipation occur and the equality sign in the Clasius-Duhem inequality (2.40) holds (Ottosen and Ristinmaa 2005, p. 552). Thus the evolution of internal variables $\dot{\boldsymbol{D}}$ and $\dot{\kappa}$ is zero and the constitutive relations are completely described by (2.39). However, dissipation occurs in irreversible case and according to (2.40) internal variables begin to evolve. Their evolution is not yet defined. In the next section we introduce the *evolution equations* which define the evolution of the internal variables in irreversible deformation.

2.2.5 Evolution of Internal Variables

Any thermomechanical process of a body is either reversible or irreversible. As stated in previous section, no dissipation occur in reversible processes. Thus internal variables do not evolve. The system can be returned to its initial state without any loss of energy over the whole process because no dissipation has occurred. The reversibility of a mechanical process is determined by the elasticity condition (2.7). If the state of the material does not reach the damage condition (2.6), the process is reversible.

Irreversibilities begin to occur when material's state reaches the boundary of the elastic domain. In other words the damage condition (2.6) is fulfilled. After that, dissipation begins and therefore internal variables start to change. After any dissipation have happened, the system cannot be returned to its original state without any extra energy supply to the system. This is due to the statement that every time internal variables change, dissipation occur.

Previously we said that the constitutive relations (2.39) and (2.40) shown in the previous section do not describe the change of internal variables. Hence we need to define the evolution equations. It is of course possible to postulate some evolution laws and check afterwards that they satisfy the dissipation inequality. However, Onsager (1931*a*,*b*) proposed a more systematic method to derive linear evolution equations. Edelen (1972) made a nonlinear extension to these *Onsagers reciprocal relations*. The generalized method is called *potential approach*.

We begin by defining *thermodynamic forces* $\boldsymbol{Y}$ and K which are dual to the rates of internal variables $\dot{\boldsymbol{D}}$ and $\dot{\kappa}$ respectively:

$$\boldsymbol{Y} \equiv \rho_0 \frac{\partial \psi^c}{\partial \boldsymbol{D}}, \qquad K \equiv -\rho_0 \frac{\partial \psi^c}{\partial \kappa}. \tag{2.41}$$

Thermodynamic forces (2.41) can be combined to a set $\boldsymbol{W}$ as

$$\boldsymbol{W} = \{\boldsymbol{Y}, K\}. \tag{2.42}$$

Similarly, internal variables can be expressed as a set $\boldsymbol{\kappa}$:

$$\boldsymbol{\kappa} = \{\boldsymbol{D}, -\kappa\}. \tag{2.43}$$

Then the dissipation inequality (2.40) can be expressed as

$$\gamma = \boldsymbol{W} : \dot{\boldsymbol{\kappa}} \geq 0. \tag{2.44}$$

Next we postulate that there exists a *dissipation potential function* φ such that it relates the thermodynamic forces and internal variable rates as

$$\dot{\boldsymbol{\kappa}} = \dot{\lambda} \frac{\partial \varphi}{\partial \boldsymbol{W}}, \tag{2.45}$$

where $\dot{\lambda}$ is a non-negative scalar multiplier which will be solved later. Dissipation potential can depend on thermodynamic forces and possibly some other variables $\boldsymbol{Z}$:

$$\varphi = \varphi(\boldsymbol{W}, \boldsymbol{Z}). \tag{2.46}$$

The direction of the evolution of internal variables $\boldsymbol{\kappa}$ can be seen as outward pointing normal of the dissipation potential surface $\varphi = 0$ illustrated in figure 2.3. The dissipation potential must be chosen such that the dissipation inequality (2.44) is satisfied. Eringen (1975) proved that the resulting evolution laws (2.45) satisfy dissipation inequality if the dissipation potential function is a convex function and the following condition holds:

$$\varphi(\boldsymbol{W}, \boldsymbol{Z}) \geq \varphi(\boldsymbol{0}, \boldsymbol{Z}). \tag{2.47}$$

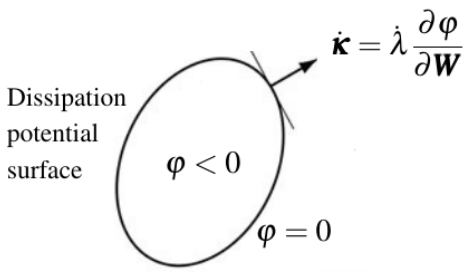


Figure 2.3. *The direction of evolution of the internal variables $\boldsymbol{\kappa}$. $\dot{\lambda}$ is a positive multiplier defined by the consistency condition.*

We are allowed to use the damage function (2.6) as the dissipation potential function. In that case we have *associated flow rule*. This assumption works good for metals and steel. If the damage function and dissipation potential function are different, we talk about *nonassociated flow rule* which works for concrete, rocks and soils. (Ottosen and Ristinmaa 2005, p. 233)

The multiplier $\dot{\lambda}$ can be determined from the *consistency condition assumption*. It requires that material's state must be retained within the elastic domain during any process. Elastic domain (2.7) is now

$$f(\boldsymbol{\sigma}, \boldsymbol{W}) \leq 0. \tag{2.48}$$

Equation (2.48) must hold even for irreversible loading. Thus the consistency condition can be expressed as

$$\dot{f}(\boldsymbol{\sigma}, \boldsymbol{W}) = 0. \tag{2.49}$$

It can be expanded using the chain rule:

$$\dot{f} = \frac{\partial f}{\partial \boldsymbol{\sigma}} : \dot{\boldsymbol{\sigma}} + \frac{\partial f}{\partial \boldsymbol{W}} : \dot{\boldsymbol{W}} = 0. \tag{2.50}$$

Because the set of termodynamic forces $\boldsymbol{W}$ consists of associated internal variables $\boldsymbol{Y}$ and K as defined in (2.42), the equation (2.50) can be expanded to the form

$$\dot{f} = \frac{\partial f}{\partial \boldsymbol{\sigma}} : \dot{\boldsymbol{\sigma}} + \frac{\partial f}{\partial \boldsymbol{Y}} : \dot{\boldsymbol{Y}} + \frac{\partial f}{\partial K}\dot{K} = 0. \tag{2.51}$$

According to postulate (2.45) the multiplier $\dot{\lambda}$ appears in the above expression and can then be solved. The evolution of individual internal variables can be calculated from (2.45) as

$$\dot{\boldsymbol{D}} = \dot{\lambda}\frac{\partial \varphi}{\partial \boldsymbol{Y}}, \qquad \dot{\kappa} = -\dot{\lambda}\frac{\partial \varphi}{\partial K} \tag{2.52}$$

and the system can then be solved also for irreversible deformations.

2.3 Specific Concrete Material Model

Now that we have derived the constitutive equations using the specific complementary Helmholtz free energy (later just *free energy*) and the dissipation potential. The elastic domain was bounded by the damage surface. In order to construct a specific material model, we need to define specific expressions for these components. In this section we will formulate the material model according to expressions proposed by Yaghoubi et al. (2014) and Hartikainen et al. (2018).

2.3.1 Free Energy

The specific free energy is composed of two parts:

$$\psi^{\mathrm{c}}(\boldsymbol{\sigma},\boldsymbol{D},\kappa) = \psi_1^{\mathrm{c}}(\boldsymbol{\sigma},\boldsymbol{D}) + \psi_2^{\mathrm{c}}(\kappa). \tag{2.53}$$

The first part $\psi_1^{\mathrm{c}}(\boldsymbol{\sigma},\boldsymbol{D})$ is a scalar-valued isotropic free energy function. It describes the reversible elastic energy stored in the system and takes the damage-degragation into account. The principle of material frame-indifference states that the following must hold:

$$\psi_1^{\mathrm{c}}(\boldsymbol{\sigma},\boldsymbol{D}) = \psi_1^{\mathrm{c}}(\boldsymbol{Q}\boldsymbol{\sigma}\boldsymbol{Q}^{\mathrm{T}},\boldsymbol{Q}\boldsymbol{D}\boldsymbol{Q}^{\mathrm{T}}), \tag{2.54}$$

where $\boldsymbol{Q}$ is a proper orthogonal second order tensor. Function $\psi_1^{\mathrm{c}}(\boldsymbol{\sigma},\boldsymbol{D})$ can be formulated by using terms received by applying the integrity basis for $\boldsymbol{\sigma}$ and $\boldsymbol{D}$:

$$\left\{\mathrm{tr}\,\boldsymbol{\sigma}, \mathrm{tr}(\boldsymbol{\sigma}^2), \mathrm{tr}(\boldsymbol{\sigma}^3), \mathrm{tr}\,\boldsymbol{D}, \mathrm{tr}(\boldsymbol{D}^2), \mathrm{tr}(\boldsymbol{D}^3), \mathrm{tr}(\boldsymbol{\sigma}\boldsymbol{D}), \mathrm{tr}(\boldsymbol{\sigma}\boldsymbol{D}^2), \mathrm{tr}(\boldsymbol{\sigma}^2\boldsymbol{D}), \mathrm{tr}(\boldsymbol{\sigma}^2\boldsymbol{D}^2)\right\}. \tag{2.55}$$

The simpliest possible form for representing damage-degragating elastic behaviour is to consider only the quadratic stress terms and linear damage terms. By neglecting higher order damage terms we lose the effect of crack interaction (Basista 2003). The specific free energy expression Yaghoubi et al. have proposed is

$$\rho_0\psi_1^{\mathrm{c}}(\boldsymbol{\sigma},\boldsymbol{D}) = \frac{1+\nu}{2E}\mathrm{tr}(\boldsymbol{\sigma}^2) - \frac{\nu}{2E}(1+\alpha_1\,\mathrm{tr}\,\boldsymbol{D})(\mathrm{tr}\,\boldsymbol{\sigma})^2 + \frac{\alpha_2}{E}\mathrm{tr}(\boldsymbol{\sigma}^2\boldsymbol{D}), \tag{2.56}$$

where material parameter E is the initial *elastic modulus* of the undamaged material and ν is corresponding *Poisson's ratio*. Material parameters α_1 and α_2 control the volume change due to damage.

The energy stored due to the hardening-softening of the material is described by the part $\psi_2^{\mathrm{c}}(\kappa)$ in equation (2.53). It is defined as:

$$\rho_0\psi_2^{\mathrm{c}}(\kappa) = \int_0^{\kappa}\int_0^{\kappa'} g(\kappa'')\mathrm{d}\kappa''\mathrm{d}\kappa'. \tag{2.57}$$

The integrand $g(\kappa)$ is a four-parameter rational function

$$g(\kappa) = \frac{H_0}{\kappa_0}\frac{a_2\left(\kappa/\kappa_0\right)^2 - 2a_1\left(\kappa/\kappa_0\right) - 1}{\left[a_2\left(\kappa/\kappa_0\right)^2 + 1\right]^2}. \tag{2.58}$$

Hardening-softening behaviour is discussed more in the next section. The characteristic shape of $g(\kappa)$ is illustrated in figure 2.4a.

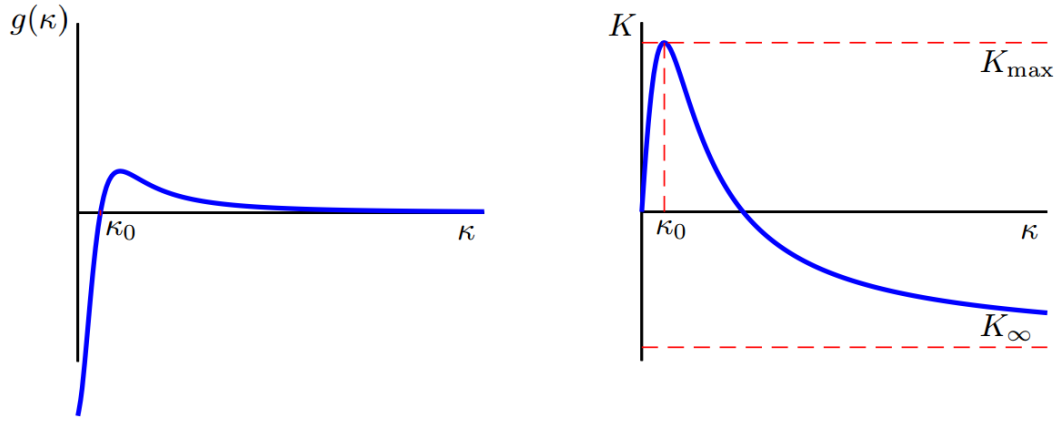


Figure 2.4. *(a) Response of material to the hardening variable κ. (b) Hardening variable K. (Hartikainen et al. 2018)*

2.3.2 Elastic Domain

This model utilizes the Ottosen's 4 parameter damage criterion presented in section 2.2.2. It is modified to take isotropic hardening-softening into account:

$$f(\boldsymbol{\sigma},\boldsymbol{Y},K) = \frac{A}{\sigma_{c0}} J_2 + \Lambda\sqrt{J_2} + BI_1 - (\sigma_{c0} + K). \tag{2.59}$$

First terms are the same as in original failure criterion (2.8). Material parameters σ_{c0}, A, B, k_1 and k_2 are defined by the initial damage surface. Note that parameters k_1 and k_2 appear inside term Λ.

Last term K is a function that controls the hardening-softening. It relates to the specific free energy via the definition $(2.41)_2$. It takes the following form:

$$K(\kappa) = \int_0^\kappa g(\kappa')\mathrm{d}\kappa' = H_0 \frac{a_1 (\kappa/\kappa_0)^2 + (\kappa/\kappa_0)}{a_2 (\kappa/\kappa_0)^2 + 1}. \tag{2.60}$$

Equation (2.60) results in the characteristic hardening-softening behaviour illustrated in figure 2.4b.

Material parameters H_0, κ_0, a_1 and a_2 are determined by assuming that when hardening variable κ approaches infinity, $K = K_\infty$ approaches negative initial compressive strength $-\sigma_{c0}$. When also the point $(\kappa_0, K_{\max})$ is calibrated to represent the ultimate uniaxial compressive stress σ_c and corresponding strain ε_c, all four material parameters are determined.

2.3.3 Dissipation Potential

The dissipation potential is defined as a function that depends on the thermodynamic forces $\boldsymbol{Y}$ and K. We neglect thermal effects and inelastic strains and also assume that the rate

of loading $\dot{\boldsymbol{\sigma}}$ has no effect on dissipation. A formulation for such dissipation potential function is proposed by Yaghoubi et al. as

$$\varphi(\boldsymbol{Y},K;\boldsymbol{\sigma}) = \mathrm{I}_\Sigma(\boldsymbol{Y},K;\boldsymbol{\sigma}). \tag{2.61}$$

It is emphasised that $\boldsymbol{\sigma}$ only acts as a parameter in the dissipation potential. The indicator function I_Σ is defined by Frémond (2002) as:

$$\mathrm{I}_\Sigma(\boldsymbol{Y},K;\boldsymbol{\sigma}) = \begin{cases} 0, & \text{if } (\boldsymbol{Y},K) \in \Sigma \\ +\infty, & \text{if } (\boldsymbol{Y},K) \notin \Sigma \end{cases}, \tag{2.62}$$

where Σ is a convex set that determines the admissible domain for thermodynamic forces $\boldsymbol{Y}$ and K. It is defined by utilizing the modified Ottosen's failure criterion:

$$\Sigma = \left\{ (\boldsymbol{Y},K) \in \mathbb{R}^6 \times \mathbb{R} | f(\boldsymbol{Y},K;\boldsymbol{\sigma}) \leq 0 \right\}. \tag{2.63}$$

The failure criterion $f(\boldsymbol{Y},K;\boldsymbol{\sigma})$ in (2.63) is a modified form of the failure surface (2.59):

$$f(\boldsymbol{Y},K;\boldsymbol{\sigma}) = \frac{A}{\sigma_{\mathrm{c0}}}\tilde{J}_2 + \Lambda\sqrt{\tilde{J}_2} + BI_1 - (\sigma_{\mathrm{c0}} + K). \tag{2.64}$$

As $f(\boldsymbol{Y},K;\boldsymbol{\sigma})$ is part of the definition of dissipation potential, now it can only depend on thermodynamic variables. Stress $\boldsymbol{\sigma}$ acts only as a parameter. The dependency on the thermodynamic variable $\boldsymbol{Y}$ is obtained by redefining the deviatoric stress invariants as

$$\tilde{J}_2 = \frac{1}{2}\left[\frac{E}{\alpha_2}\operatorname{tr}\boldsymbol{Y} + (\operatorname{tr}\boldsymbol{\sigma})^2\left(\frac{3\alpha_1\nu}{2\alpha_2} - \frac{1}{3}\right)\right], \tag{2.65}$$

$$\tilde{J}_3 = \frac{1}{3}\left[\frac{E}{\alpha_2}(\operatorname{tr}(\boldsymbol{\sigma}\boldsymbol{Y}) - (\operatorname{tr}\boldsymbol{\sigma})(\operatorname{tr}\boldsymbol{Y})) + \left(\frac{2}{9} - \frac{\alpha_1\nu}{\alpha_2}\right)(\operatorname{tr}\boldsymbol{\sigma})^3\right]. \tag{2.66}$$

Lode angle must also be redefined as

$$\cos 3\theta = \frac{3\sqrt{3}}{2}\frac{\tilde{J}_3}{\tilde{J}_2{}^{3/2}}. \tag{2.67}$$

Expressions (2.65) and (2.66) are derived using relations $\operatorname{tr}(\boldsymbol{\sigma}^2) = 2J_2 + \frac{1}{3}I_1^2$, $\operatorname{tr}(\boldsymbol{\sigma}^3) = 3J_3 + \operatorname{tr}(\boldsymbol{\sigma}^2)I_1 - \frac{2}{9}I_1^3$ and expression (2.71) that defines the thermodynamic force $\boldsymbol{Y}$.

The subgradients $\partial\varphi_{\mathrm{Y}}$ and $\partial\varphi_{\mathrm{K}}$ of dissipation potential control the evolution of internal variables $\boldsymbol{D}$ and κ as is defined in section 2.2.5. They are defined only in the admissible domain so that

$$\partial\varphi(\boldsymbol{Y},K;\boldsymbol{\sigma}) = \begin{cases} \{(\partial\varphi_{\mathrm{Y}}, \partial\varphi_{\mathrm{K}})\}, & \text{if } (\boldsymbol{Y},K) \in \Sigma \\ \emptyset, & \text{if } (\boldsymbol{Y},K) \notin \Sigma \end{cases}. \tag{2.68}$$

Inside the admissible region Σ the subgradients are defined as zeros. At the boundary of the admissible domain they are normals to the damage surface $f(\boldsymbol{Y},K;\boldsymbol{\sigma})$:

$$(\partial\varphi_{\mathrm{Y}}, \partial\varphi_{\mathrm{K}}) = \begin{cases} (\boldsymbol{0},0), & \text{if } f(\boldsymbol{Y},K;\boldsymbol{\sigma}) < 0 \\ \left(\dfrac{\partial f}{\partial \boldsymbol{Y}}, \dfrac{\partial f}{\partial K}\right), & \text{if } f(\boldsymbol{Y},K;\boldsymbol{\sigma}) = 0 \end{cases}. \tag{2.69}$$

The dissipation potential is a modified form of the Ottosen's convex damage surface. It is also subdifferentiable and fullfills the condition (2.47). Subgradients $\partial\varphi_{\mathbf{Y}}$ and $\partial\varphi_{\mathrm{K}}$ belong to the dual space of the thermodynamic forces $\{\boldsymbol{Y}, K\}$. Therefore the proposed dissipation potential is thermodynamically admissible.

2.3.4 Constitutive Equations

Now that the expressions for free energy, elastic domain and the dissipation potential are known, we can derive the constitutive relations using equations (2.39), (2.41) and (2.52). They result in the following constitutive equations:

$$\boldsymbol{\varepsilon} = \rho_0 \frac{\partial \psi^{\mathrm{c}}}{\partial \boldsymbol{\sigma}} = \frac{1+\nu}{E}\boldsymbol{\sigma} - \frac{\nu}{E}(1+\alpha_1 \operatorname{tr}\boldsymbol{D})(\operatorname{tr}\boldsymbol{\sigma})\boldsymbol{I} + \frac{\alpha_2}{E}(\boldsymbol{\sigma}\boldsymbol{D}+\boldsymbol{D}\boldsymbol{\sigma}), \tag{2.70}$$

$$\boldsymbol{Y} = \rho_0 \frac{\partial \psi^{\mathrm{c}}}{\partial \boldsymbol{D}} = -\frac{\alpha_1 \nu}{2E}(\operatorname{tr}\boldsymbol{\sigma})^2\boldsymbol{I} + \frac{\alpha_2}{E}\boldsymbol{\sigma}^2, \tag{2.71}$$

$$\dot{\boldsymbol{D}} = \dot{\lambda}\frac{\partial f}{\partial \boldsymbol{Y}} = \dot{\lambda}\left[\left(\frac{A}{\sigma_{\mathrm{c0}}} + \sqrt{\tilde{J}_2}\frac{\partial \Lambda}{\partial \tilde{J}_2} + \frac{\Lambda}{2\sqrt{\tilde{J}_2}}\right)\frac{\partial \tilde{J}_2}{\partial \boldsymbol{Y}} + \sqrt{\tilde{J}_2}\frac{\partial \Lambda}{\partial \tilde{J}_3}\frac{\partial \tilde{J}_3}{\partial \boldsymbol{Y}}\right], \tag{2.72}$$

$$K = -\rho_0 \frac{\partial \psi^{\mathrm{c}}}{\partial \kappa} = H_0 \frac{a_1 (\kappa/\kappa_0)^2 + (\kappa/\kappa_0)}{a_2 (\kappa/\kappa_0)^2 + 1}, \tag{2.73}$$

$$\dot{\kappa} = -\dot{\lambda}\frac{\partial f}{\partial K} = \dot{\lambda}. \tag{2.74}$$

Multiplier $\dot{\lambda}$ can be solved from the consistency condition (2.51). It is zero for reversible behaviour and positive if irreversibilities occur.

2.3.5 Material Parameters

The model has 13 material parameters. They are related to actual material behaviour in following manner:

- K_{b} and G: elastic behaviour of the material
- A, B, k_1, k_2 and σ_{c0}: initial damage surface
- H_0, κ_0, a_1 and a_2: hardening-softening behaviour
- α_1 and α_2: volume dilatation

Values for these parameters are determined from experimental findings by Kupfer et al. (1969). They have tested concrete specimens with an ultimate uniaxial compression strength of 32.8 MPa. Elastic modulus E and Poisson's ratio ν are determined from figures 5.1a and 5.2a that are presented in chapter 5:

$$E = 32\,\mathrm{GPa}, \qquad \nu = 0.2. \tag{2.75}$$

Initial damage surface is determined by following assumptions based on the experimental data: $\sigma_{c0} = 18\,\text{MPa}$, $\sigma_{t0}/\sigma_{c0} = 0.09$, $\sigma_{bc0}/\sigma_{c0} = 1.16$ and a point on the compressive meridian $(\sigma_{m4}, \sigma_{e4}) = (-2.89\sigma_{c0}, 4.90\sigma_{c0})$. Insertion to equations (2.8) – (2.10) yields material parameters

$$A = 2.978, \quad B = 6.078, \quad k_1 = 20.56, \quad k_2 = 0.999, \quad \sigma_{c0} = 18\,\text{MPa}. \tag{2.76}$$

Hardening-softening–related parameters are adjusted so that the point (κ_0, K_{max}) in hardening-softening graph (figure 2.4b) corresponds to the experimental ultimate uniaxial compressive stress state $(\sigma_c, \varepsilon_c) = (-2.887\sigma_{c0},\ 4.899\sigma_{c0})$. This can be ensured by adjusting the parameters in equation (2.60). Also the condition for K_∞ introduced in section 2.3.2 has been accounted. Following parameters satisfy these conditions:

$$H_0 = 85.3\,\text{MPa}, \qquad \kappa_0 = 4.41 \cdot 10^{-6}, \qquad a_1 = -0.148, \qquad a_2 = 0.703. \tag{2.77}$$

Material parameters α_1 and α_2 are adjusted to represent volume dilation in uniaxial compression. Determined values are

$$\alpha_1 = 5.2 \qquad \text{and} \qquad \alpha_2 = 10. \tag{2.78}$$

3. NONLINEAR MATERIAL MODELS IN ELMER

3.1 Elmer FE Software

Elmer is an open source Finite Element (FE) software package. Its development begun in 1995 as a part of Finnish national computational fluid dynamics technology program. After the project ended, CSC – IT Center for Science continued to develop Elmer in different application fields. In 2005 Elmer was released under an open source license GNU General Public License. Elmer software package can be downloaded for free at CSC's website (see *Overview of Elmer,* Råback and Malinen 2018).

Elmer is meant for solving partial differential equations utilizing Finite Element Method. Problems can be purely mathematical or related to some physical phenomena. Standard Elmer distribution contains solution packages for several fields including structural mechanics, fluid dynamics, heat transfer, acoustics, etc. Elmer also offers a variety of weak couplings between these fields. Therefore it is well suited for multiphysical simulations. However, because it is a free open-source software, it lacks some sophisticated features in individual fields. But for the same reason it is also relatively easy to program new solver packages to suit user's needs.

Elmer software package does not contain comprehensive mesh generation or postprocessing tools. Although ElmerGrid is able to generate simple 2-dimensional (2D) or 3-dimensional (3D) meshes, some external mesh generator is recommended such as Gmsh or SALOME. Elmer does not contain a geometry generation tool, either. For postprocessing Elmer has two basic built-in tools; ElmerPost and VTK widget of ElmerGUI. ParaView is recommended for more sophisticated visualization. Elmer also provides tools to export data in text format. Elmer features a configurable graphical user interface.

Even though Elmer is not a dedicated structural mechanics analysis software, there are quite a lot of features already available for mechanical analysis. Elmer's mechanical solvers are separated onto 4 different modules. Two for linear problems and two for nonlinear problems. In addition to that, there exist some general techniques that can be applied in mechanics problems. Documentation can be found in *Elmer Models Manual* and *Elmer Solver Manual* (Råback et al. 2018; Ruokolainen et al. 2018).

Elmer's solver module *StressSolve* can perform following linear structural analysis:

- Static problems
- Transient problems
- Modal problems
- Harmonic problems
- Stability problems.

Most of these analysis accept linear anisotropic material models, viscous or Rayleigh damping, prestess, prestrain and thermal stresses. Modal analysis is able to take geometric stiffness into account. Geometry must be 3D or 2D in plane stress or plane strain state. Also cylindrically symmetric 2D simulations are supported. Model lumping can be performed.

Elmer also features a 2D linear elastic plate solver module *Smitc*. It is based on the Reissner-Mindlin plate model. It accepts linear isotropic material model and includes hole correction. Static, transient and modal analysis is supported. Viscous damping and prestress can be introduced.

Nonlinear structural problems are handled by *ElasticSolve* module. It can solve static and transient problems in full 3D geometry or 2D plane stress and plane strain states. Solver accepts linear isotropic or anisotropic material models. It also has built-in compressible or incompressible Neo-Hookean material model. In addition to these material models, constitutive relations can be defined via *user defined material subroutine* (UMAT). So far it works only for geometrically linear static problems. Elmer's UMAT subroutine is discussed in-depth in chapter 4.

Module *ShellSolver* can solve nonlinear shell problems. It is used for solving thin elastic plate problems for small or large displacements. At the time of writing this thesis the module is fairly new and will probably have more features in the future.

There are some primitive methods to solve simple contact mechanics via mortar boundary conditions. Mortar conditions can be applied on any elasticity equation that rely on solving the displacements at each node. This feature is also still under development and not even documented properly yet.

As seen above, Elmer can simulate only 2 types of materials; Hookean and Neo-Hookean. The objective of this thesis is to add a previously formulated continuum damage material model into Elmer by utilizing the UMAT-interface. This allows the steady state simulation of quasi-brittle materials in Elmer.

3.2 FE Discretization

As we saw in previous chapter, the governing equations are partial differential equations. Analytical solutions can exist for some simplier problems, but in order to solve more

complex problems, we need to apply numerical methods. Finite Element Method is one widely used example of such method. As Elmer is based on FEM, it is explained here to the extent what is necessary for this thesis. FEM is widely studied method and thus numerous textbooks exist on the topic. This chapter is mainly based on a book written by Wriggers (2008).

In static structural analysis the equilibrium equation is solved for a finite body B. When the displacement field $\boldsymbol{u}$ and therefore also strain field $\boldsymbol{\varepsilon}$ is considered as unknown quantities, the variational form of equilibrium equation in *Voigt notation* becomes

$$\int_B \boldsymbol{\sigma} \cdot \delta\boldsymbol{\varepsilon}\, dV - \int_B \boldsymbol{b} \cdot \delta\boldsymbol{u}\, dV - \int_{\partial B} \boldsymbol{t} \cdot \delta\boldsymbol{u}\, dA = 0. \tag{3.1}$$

The first term in equation (3.1) denotes the internal forces of the body, second term is external body forces acting on the body and third term is external tractions acting on the surface ∂B of the body. The equation can then solved using finite element method. As the displacement field was considered an unknown quantity, our task is to solve it. Therefore FEM is *displacement method*.

Body B is arbitrary, so its geometry can be complicated. Thus the arbitrary domain needs to be discretized to smaller individual finite elements. Every element has a certain number of nodes. The solution fields are interpolated element-wise, gathered to the element nodes and assembled globally to form an approximation of the original computational domain. The accuracy of the solution depends on the computational mesh used. A good computational mesh represents well the geometry of original domain and is fine enough so that the piecewise defined solution fields describe adequately the real solution to the problem. Mesh cannot contain infinitely many elements as their amount roughly corresponds to the computational complexity.

The computational domain can be divided into certain types of simplier geometric elements, the *reference elements*. The type of elements that we can use depends on the mesh generator and FE-solver. According to Elmer Solver Manual (Ruokolainen et al. 2018, Appendix D) currently supported element types in Elmer are:

- 0D nodal element (1 node)
- 1D line segment (2, 3 or 4 nodes)
- 2D triangle (3, 6 or 10 nodes)
- 2D quadrilateral (4, 8, 9, 12 or 16 nodes)
- 3D tetrahedron (4 or 10 nodes)
- 3D pyramid (5 or 13 nodes)
- 3D wedge (6 or 15 nodes)
- 3D hexahedron (8, 20 or 27 nodes)

The above elements in their reference configuration are geometrically the simpliest possible representations of their type. For example 2-dimensional quadrilateral element is a straight edged square placed in the reference coordinate axes in the most simpliest position, orientation and size possible. Reference elements have their own local coordinate system. Real elements in the discretized computation mesh do not need to be that simple. Reference elements can be placed anywhere in the computational domain, oriented in any direction and their shape can be distorted to form the full computational mesh. Of course there are some limitations to the transformation from reference to real element. The most obvious is that the transformation must preserve the amount of nodes and boundaries. The transformation between real elements and reference elements is described mathematically by applying the Jacobian matrix $\boldsymbol{J}_{\mathrm{e}}$ between global coordinates $\boldsymbol{x}$ and local reference element coordinates $\boldsymbol{\xi}$. Transformation is illustrated in figure 3.1.

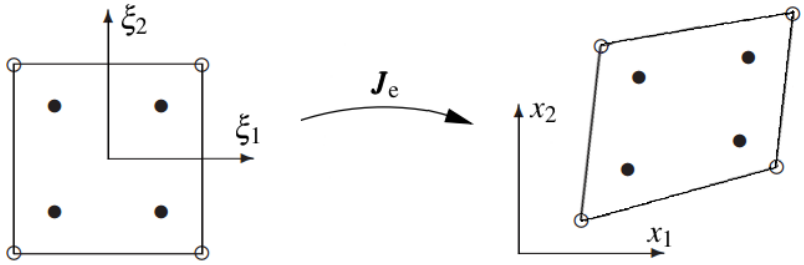


Figure 3.1. *Transformation between 2D quadrilateral reference element (left) and actual element (right). Black dots indicate Gaussian integration point positions.*

Now that we have partitioned the problem to smaller and simplier geometrical elements, we need to express the unknown displacement field inside each element. This can be done by introducing *Ansatz shape functions* $N_{\mathrm{i}}(\boldsymbol{\xi})$ to interpolate the displacement field $\boldsymbol{u}(\boldsymbol{\xi})$ inside each element. Shape functions relate the nodal displacements $\boldsymbol{q}$ and displacement field as

$$\boldsymbol{u}(\boldsymbol{\xi}) = \sum_{\mathrm{i}=1}^{\mathrm{n}} N_{\mathrm{i}}(\boldsymbol{\xi})\boldsymbol{q}_{\mathrm{i}}(\boldsymbol{\xi}), \tag{3.2}$$

where n is the number of shape functions. In *isoparametric* element formulations n is also the number of element nodes. We can shorten above expression by unifying node and shape function ordering to obtain

$$\boldsymbol{u} = \boldsymbol{N}\boldsymbol{q}. \tag{3.3}$$

The strain field $\boldsymbol{\varepsilon}(\boldsymbol{\xi})$ is obtained by differentiating expression (3.2) with respect to the global coordinates $\boldsymbol{x}$:

$$\boldsymbol{\varepsilon}(\boldsymbol{\xi}) = \sum_{\mathrm{i}=1}^{\mathrm{n}} \frac{\partial N_{\mathrm{i}}(\boldsymbol{\xi})}{\partial \boldsymbol{x}} \boldsymbol{q}_{\mathrm{i}}(\boldsymbol{\xi}) = \sum_{\mathrm{i}=1}^{\mathrm{n}} \boldsymbol{B}_{\mathrm{i}}(\boldsymbol{\xi})\boldsymbol{q}_{\mathrm{i}}(\boldsymbol{\xi}), \tag{3.4}$$

where $\boldsymbol{B}_\mathrm{i}$ is the part of *kinematic matrix* that is related to shape function i. Again, we shorten the expression by unifying node and shape function ordering:

$$\boldsymbol{\varepsilon} = \boldsymbol{B}\boldsymbol{q}. \tag{3.5}$$

Now $\boldsymbol{B}$ is kinematic matrix for the whole element. Variational forms of $\boldsymbol{u}$ and $\boldsymbol{\varepsilon}$ are

$$\delta\boldsymbol{u} = \boldsymbol{N}\delta\boldsymbol{q}, \qquad \delta\boldsymbol{\varepsilon} = \boldsymbol{B}\delta\boldsymbol{q}. \tag{3.6}$$

Inserting them to variational form of equilibrium equation (3.1) and realizing that the equation must hold for any arbitrary displacement variation $\delta\boldsymbol{q}$ we obtain

$$\int_B \boldsymbol{B}^\mathrm{T}\boldsymbol{\sigma}\, dV - \int_B \boldsymbol{N}^\mathrm{T}\boldsymbol{b}\, dV - \int_{\partial B} \boldsymbol{N}^\mathrm{T}\boldsymbol{t}\, dA = 0. \tag{3.7}$$

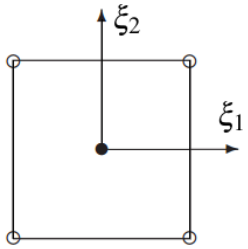
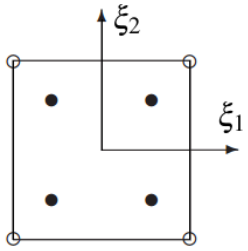
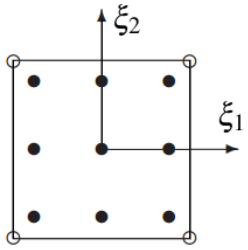
Field variables in equation (3.7) are integrated over the computational domain B. As we have divided the whole domain into elements, we need to integrate the continuum field variables over individual reference elements. As we solve the system nodal-wise, we need to gather the information to element's nodes. This can be done by applying the numerical *Gauss-Legendre* integration. The basic idea for integrating any general continuum property $f(\boldsymbol{x})$ over the element B_e is to evaluate it's value in specified points inside reference element, weigh it by factor W_p and transform to the global coordinate system:

$$\int_{B_\mathrm{e}} f(\boldsymbol{x})\, dV \approx \sum_{\mathrm{p}=1}^{\mathrm{n_p}} f(\boldsymbol{\xi}_\mathrm{p})\, W_\mathrm{p}\, \det[\boldsymbol{J}_\mathrm{e}(\boldsymbol{\xi}_\mathrm{p})]. \tag{3.8}$$

Equation (3.8) is presented as a volume integral, but it can be applied to lower dimensional elements, too. Weight factors W_p and local coordinates $\boldsymbol{\xi}_\mathrm{p}$ are usually defined by utilizing polynomial approximations. Values for 2-dimensional quadrilateral element is given in table 3.1 as an example. Column m indicates the order of polynomial approximation. File *elements.def* supplied with the Elmer distribution contains these values for every supported element type (Elmer Solver Manual, Ruokolainen et al. 2018, Appendix D).

This kind of procedure allows us to evaluate material response only in certain points. As we rely on the principle of local action (see section 2.2.1), material response can be evaluated independently from the nearby material points.

Table 3.1. *Gauss integration points for 2-dimensional quadrilateral element. Adapted from (Wriggers 2008, p. 117).*

m	n_p	p	ξ_{1p}	ξ_{2p}	W_p	**Position of points**
1	1	1	0	0	4	
3	4	1	$-1/\sqrt{3}$	$-1/\sqrt{3}$	1	
		2	$+1/\sqrt{3}$	$-1/\sqrt{3}$	1	
		3	$-1/\sqrt{3}$	$+1/\sqrt{3}$	1	
		4	$+1/\sqrt{3}$	$+1/\sqrt{3}$	1	
5	9	1	$-\sqrt{3/5}$	$-\sqrt{3/5}$	25/81	
		2	0	$-\sqrt{3/5}$	40/81	
		3	$+\sqrt{3/5}$	$-\sqrt{3/5}$	25/81	
		4	$-\sqrt{3/5}$	0	40/81	
		5	0	0	64/81	
		6	$+\sqrt{3/5}$	0	40/81	
		7	$-\sqrt{3/5}$	$+\sqrt{3/5}$	25/81	
		8	0	$+\sqrt{3/5}$	40/81	
		9	$+\sqrt{3/5}$	$+\sqrt{3/5}$	25/81	

3.3 Sources of Nonlinearity in Structural Modelling

Generally every nonlinear phenomenon that we want to simulate causes nonlinearities to the FE-model. As computers can only perform linear algebraic operations, presence of any nonlinearity requires that the system must be solved using special techniques. These solution algorithms are discussed more in the next section. Nonlinearities in static structural analysis are mainly caused by

- Nonlinear constitutive relations
- Geometrical nonlinearities
- Nonlinear boundary conditions

Nonlinear constitutive relations have already been discussed in section 2.1.

Geometrical nonlinearities occur when relatively large displacements are considered. Common structural materials such as steel or concrete can handle only relatively small strains before failure. For example the concrete type discussed in chapter 5 has ultimate compressive strain 0.21 % and ultimate tensile strain of only 0.091 %. Therefore we have accepted that the deformations are quite small in order to simplify the model. Neglecting the nonlinear geometry effects makes the FE-problem less nonlinear and thus it should be less computationally expensive.

The definition of *small displacements* is vague. In general, there are no definite rules to determine whether linear geometry assumption is valid or not. In addition to modelling large displacements, some phenomenon require nonlinear geometrical model. Example of such case is stability problems, where buckling can occur at seemingly small displacements.

If real structures are simulated accurately, boundary conditions often posses a source of nonlinearity. The very fundamental cause of nonlinearity in boundary conditions arises from detecting whether two surfaces are touching or not. If there is a gap, surfaces do not transmit traction forces between them. But when the surfaces contact, they begin to transmit some amount of contact forces as they can't penetrate inside each others. Moreover, when surfaces are in contact, contact area depends on the applied load.

Contact mechanics is a whole branch of physics. Numerical treatment of contact problems requires special contact elements in the contact region. Formulating such elements is time consuming. Furthermore, contact mechanics mainly deal with the boundaries of solid bodies whereas material modelling is more related to the domain itself. Thus contact nonlinearities are not in the scope of this thesis.

3.4 Nonlinear Solution Methods

In previous sections we reduced the general partial differential equation system to an algebraic equation system by utilizing FEM. Due to nonlinearities introduced in previous section, the system is nonlinear. Hence we need a method to solve these equations.

Most common methods linearize the problem to advance from one equilibrium solution to the next one. In static structural problems usually the first equilibrium state is stress-free state with no displacements. Then the loads are applied in increments until the desired loading is achieved. *Explicit* solution methods assume the current state as a valid solution, advances some increment forwards based on the current state and assumes that the achieved solution is valid as well. *Implicit* methods are similar, but the new state is evaluated using the new, yet unknown state as well. Usually instead of accepting the new state as a valid solution straight away, implicit methods can have some method to verify and improve the solution until sufficient convergence is achieved. Elmer uses implicit *Newton-Raphson* method as default.

In order to utilize these methods, the problem needs to be linearized. The linearization is

done by applying the *Taylor series expansion*. In general format it can be expressed as

$$f(\boldsymbol{x}_2) \approx f(\boldsymbol{x}_1) + \frac{\partial f(\boldsymbol{x}_1)}{\partial \boldsymbol{x}} \cdot d\boldsymbol{x}, \tag{3.9}$$

where the increment $d\boldsymbol{x} = \boldsymbol{x}_2 - \boldsymbol{x}_1$. This equation approximates the function f value from state $\boldsymbol{x}_1$ to state $\boldsymbol{x}_2$. The term $\partial f(\boldsymbol{x})/\partial \boldsymbol{x}$ is the linearized incremental change that we need in order to utilize linearization methods.

In the static structural analysis we need an expression for linearized internal stress increment that appears in equilibrium equation (3.7). Differentiation yields

$$\frac{\partial}{\partial \boldsymbol{x}} \int_B \boldsymbol{B}^{\mathrm{T}} \boldsymbol{\sigma} \, dV = \int_B \boldsymbol{B}^{\mathrm{T}} \frac{\partial \boldsymbol{\sigma}}{\partial \boldsymbol{x}} \, dV + \int_B \frac{\partial \boldsymbol{B}^{\mathrm{T}}}{\partial \boldsymbol{x}} \boldsymbol{\sigma} \, dV \tag{3.10}$$

In case of linear geometry the last term tends to zero. The stress term depends on the strains, which further depend on the nodal displacements. The relation can be written as $\boldsymbol{\sigma} = \boldsymbol{\sigma}(\boldsymbol{\varepsilon}(\boldsymbol{q}))$. Insertion to expression (3.10) and applying chain rule yields

$$\int_B \boldsymbol{B}^{\mathrm{T}} \frac{\partial \boldsymbol{\sigma}(\boldsymbol{\varepsilon}(\boldsymbol{q}))}{\partial \boldsymbol{x}} \, dV = \int_B \boldsymbol{B}^{\mathrm{T}} \frac{\partial \boldsymbol{\sigma}}{\partial \boldsymbol{\varepsilon}} \frac{\partial \boldsymbol{\varepsilon}}{\partial \boldsymbol{q}} \frac{\partial \boldsymbol{q}}{\partial \boldsymbol{x}} \, dV = \int_B \boldsymbol{B}^{\mathrm{T}} \frac{\partial \boldsymbol{\sigma}}{\partial \boldsymbol{\varepsilon}} \boldsymbol{B} \, dV \frac{d\boldsymbol{q}}{d\boldsymbol{x}}. \tag{3.11}$$

Multiplying the equation with the displacement increment $d\boldsymbol{x}$ yields

$$\int_B \boldsymbol{B}^{\mathrm{T}} \frac{\partial \boldsymbol{\sigma}}{\partial \boldsymbol{\varepsilon}} \boldsymbol{B} \, dV \, \frac{d\boldsymbol{q}}{d\boldsymbol{x}} d\boldsymbol{x}, \tag{3.12}$$

where part $(d\boldsymbol{q}/d\boldsymbol{x})d\boldsymbol{x}$ is the incremental displacement change. Therefore the integral in equation (3.12) can be denoted as the *linearized tangential stiffness matrix* $\boldsymbol{K}_{\mathrm{T}}$ for linear geometry:

$$\boldsymbol{K}_{\mathrm{T}} = \int_B \boldsymbol{B}^{\mathrm{T}} \frac{\partial \boldsymbol{\sigma}}{\partial \boldsymbol{\varepsilon}} \boldsymbol{B} \, dV, \tag{3.13}$$

where the stress-strain relation can be nonlinear.

The global stiffness matrix is assembled from individual integration points by utilizing the Gauss-Legendre integration as described in previous section. Also the internal force vector for previous accepted equilibrium solution and for current solution iterate is assembled in similar manner. Then some linearization solution algorithm, Newton-Raphson for example, can be applied to get the next equilibrium solution.

3.5 User Material Model Procedure

In previous sections the nonlinear equilibrium equation was solved using FEM together with linearization method. As the geometry was assumed to be linear and nonlinear contact mechanics is not discussed, the nonlinearities arise solely from the constitutive relations. Stress and constitutive material Jacobian matrix can have nonlinear dependency with respect to displacement field $\boldsymbol{u}$. Different kind of material nonlinearities were discussed in section (2.1).

Various kinds of material models are readily available in structural FE solvers. Elmer's mechanical solvers have built-in linear elasticity and Neo-Hookean material models. Commercial FE-software have even more material models built into the software. However, through an *user material subroutine* (UMAT), FE-software are able to accept constitutive relations not built into the FE-code by the developers.

The primary objective of UMAT subroutine to provide updated stress vector $\boldsymbol{\sigma}$ and incremental material Jacobian $\partial\Delta\boldsymbol{\sigma}/\partial\Delta\boldsymbol{\varepsilon}$ for given strain increment $\Delta\boldsymbol{\varepsilon}$ in Gauss-Legendre integration points. FE-program supplies various input parameters describing the state of the material point. As the material response depends on the load history, updated state variables $\boldsymbol{D}$ and κ have to be supplied, too.

The workflow of FE solver using UMAT-subroutine is

1. FE-software calls UMAT-subroutine and provides all input information in the previous accepted equilibrium state except the strain increment, which is the difference in strain state from last accepted equilibrium state to current solution iterate
2. UMAT subroutine uses the input information to update the stress and material state variables and calculates the material Jacobian. Energy variables are also updated if necessary for postprocessing.
3. FE-software assembles global system of equations based on the information supplied by UMAT
4. FE-software solves the system to get new iterate for the strain increment
5. These steps are repeated until acceptable equilibrium solution is achieved.

When the procedure is completed, solution is accepted as new equilibrium state, and the procedure is repeated for the next load step.

4. IMPLEMENTING THE MATERIAL MODEL INTO ELMER

4.1 Solution Algorithm for Constitutive Equations

The main objective of mechanical constitutive modelling is to solve stress-strain relations. Also the incremental material Jacobian $\partial\Delta\boldsymbol{\sigma}/\partial\Delta\boldsymbol{\varepsilon}$ has to be supplied in order to solve the global problem. Furthermore, because the model utilizes evolution equations, material state variables need to be updated.

This is done by solving constitutive equations (2.70) – (2.74). They are solved numerically by applying Newton-Raphson iteration in the following solution algorithm. Algorithm is implemented inside the UMAT subroutine program code. It has been divided into 2 main sub-subroutines *UPDAT* and *NEWT*. Some smaller tasks are further divided into various sub-subroutines. UMAT source code can be found in appendix B.

1. Increase the strain value by the strain increment $\Delta\boldsymbol{\varepsilon}$ and compute elastic trial stress state using damage values from previous converged solution. [1]

$$\boldsymbol{\varepsilon}_{\mathrm{n+1}} = \boldsymbol{\varepsilon}_{\mathrm{n}} + \Delta\boldsymbol{\varepsilon} \tag{4.1}$$

$$\boldsymbol{\sigma}^{\mathrm{trial}}_{\mathrm{n+1}} = \boldsymbol{C}(\boldsymbol{D}_{\mathrm{n}})\boldsymbol{\varepsilon}_{\mathrm{n+1}} \tag{4.2}$$

$$\boldsymbol{Y}^{\mathrm{trial}}_{\mathrm{n+1}} = -\frac{\alpha_1 \nu}{2E}(\mathrm{tr}\,\boldsymbol{\sigma}^{\mathrm{trial}}_{\mathrm{n+1}})^2\boldsymbol{I} + \frac{\alpha_2}{E}(\boldsymbol{\sigma}^{\mathrm{trial}}_{\mathrm{n+1}})^2 \tag{4.3}$$

   Subscripts n correspond to step numbers.

2. Check the damage condition

$$f^{\mathrm{trial}}_{\mathrm{n+1}} = \frac{A}{\sigma_{\mathrm{c0}}}J_2 + \Lambda\sqrt{J_2} + BI_1 - (\sigma_{\mathrm{c0}} + K) \tag{4.4}$$

   using trial stresses calculated in step 1.

   IF $f^{\mathrm{trial}}_{\mathrm{n+1}} \leq 0$ THEN
   SET

$$\boldsymbol{\sigma}_{\mathrm{n+1}} = \boldsymbol{\sigma}^{\mathrm{trial}}_{\mathrm{n+1}} \tag{4.5}$$

$$\frac{\partial\Delta\boldsymbol{\sigma}}{\partial\Delta\boldsymbol{\varepsilon}} = \boldsymbol{C}(\boldsymbol{D}_{\mathrm{n}}) \tag{4.6}$$

   EXIT SUBROUTINE UMAT
   END IF

[1] Elasticity tensor $\boldsymbol{C}(\boldsymbol{D})$ defined in (4.18).

3. Solve $f(\lambda_{n+1}) = 0$ using iterative Newton-Raphson method.
   DO WHILE $|f(\lambda_{n+1})| > tol$, where tol is convergence tolerance value.
   - Compute iterative change $\delta\lambda$ [2]

$$f(\lambda^{i}_{n+1}) + f'(\lambda^{i}_{n+1})\delta\lambda = 0 \tag{4.7}$$
$$f(\lambda^{i}_{n+1}) = f(\boldsymbol{\sigma}^{i}_{n+1}, \boldsymbol{Y}^{i}_{n+1}, K^{i}_{n+1}) \tag{4.8}$$
$$\delta\lambda = -\frac{f(\lambda^{i}_{n+1})}{f'(\lambda^{i}_{n+1})} \tag{4.9}$$

     Superscripts i correspond to iteration numbers.
   - Update state variables $\boldsymbol{D}$ and κ

$$\boldsymbol{D}^{i+1}_{n+1} = \boldsymbol{D}^{i}_{n+1} + \delta\lambda \frac{\partial f^{i}_{n+1}}{\partial \boldsymbol{Y}^{i}_{n+1}} \tag{4.10}$$
$$\kappa^{i+1}_{n+1} = \kappa^{i}_{n+1} + \delta\lambda \tag{4.11}$$

   - Update $\boldsymbol{\sigma}$

$$\boldsymbol{\sigma}^{i+1}_{n+1} = \boldsymbol{C}(\boldsymbol{D}^{i+1}_{n+1})\boldsymbol{\varepsilon}_{n+1} \tag{4.12}$$

   END DO
   SET [3]

$$\boldsymbol{\sigma}_{n+1} = \boldsymbol{\sigma}^{i+1}_{n+1} \tag{4.13}$$
$$\boldsymbol{D}_{n+1} = \boldsymbol{D}^{i+1}_{n+1} \tag{4.14}$$
$$\kappa_{n+1} = \kappa^{i+1}_{n+1} \tag{4.15}$$
$$\frac{\partial \Delta\boldsymbol{\sigma}}{\partial \Delta\boldsymbol{\varepsilon}} = \boldsymbol{C}_{ATS}(\boldsymbol{\sigma}_{n+1}, \boldsymbol{D}_{n+1}, \kappa_{n+1}) \tag{4.16}$$

   EXIT SUBROUTINE UMAT

Equations (4.2) and (4.12) can be solved using Voigt notations for stress and strain tensors:

$$\boldsymbol{\sigma} = (\sigma_{11}, \sigma_{22}, \sigma_{33}, \sigma_{12}, \sigma_{13}, \sigma_{23})^{T}, \quad \boldsymbol{\varepsilon} = (\varepsilon_{11}, \varepsilon_{22}, \varepsilon_{33}, 2\varepsilon_{12}, 2\varepsilon_{13}, 2\varepsilon_{23})^{T}. \tag{4.17}$$

Then the elasticity tensor $\boldsymbol{C}$ is the inverse of the compliance tensor $\boldsymbol{L}$

$$\boldsymbol{L} = \begin{bmatrix} L_{11} & -\eta_2 & -\eta_2 & 2\eta_3 D_{12} & 2\eta_3 D_{13} & 0 \\ & L_{22} & -\eta_2 & 2\eta_3 D_{12} & 0 & 2\eta_3 D_{23} \\ & & L_{33} & 0 & 2\eta_3 D_{13} & 2\eta_3 D_{23} \\ & & & 2L_{44} & 2\eta_3 D_{23} & 2\eta_3 D_{13} \\ & \text{Sym.} & & & 2L_{55} & 2\eta_3 D_{12} \\ & & & & & 2L_{66} \end{bmatrix}, \tag{4.18}$$

[2] $f(\lambda^{i}_{n+1})$ is calculated using expression (2.64) and $f'(\lambda^{i}_{n+1})$ is defined in (4.21).
[3] $\boldsymbol{C}_{ATS}$ defined in (4.22).

where

$$\begin{aligned} L_{11} &= \eta_1 - \eta_2 + 2\eta_3 D_{11}, \quad & L_{22} &= \eta_1 - \eta_2 + 2\eta_3 D_{22}, \\ L_{33} &= \eta_1 - \eta_2 + 2\eta_3 D_{33}, \quad & L_{44} &= \eta_1 + \eta_3 (D_{11} + D_{22}), \\ L_{55} &= \eta_1 + \eta_3 (D_{11} + D_{33}), \quad & L_{66} &= \eta_1 + \eta_3 (D_{22} + D_{33}) \end{aligned} \tag{4.19}$$

and

$$\eta_1 = \frac{1+\nu}{E}, \qquad \eta_2 = \frac{\nu(1+\alpha_1 \operatorname{tr} \boldsymbol{D})}{E}, \qquad \eta_3 = \frac{\alpha_2}{E}. \tag{4.20}$$

In order to solve the iterative change $\delta\lambda$ in step 3 we need the expression for derivative $f'(\lambda)$:

$$\begin{aligned} f'(\lambda) = -\frac{\partial f}{\partial \boldsymbol{Y}} : \rho_0 \frac{\partial^2 \psi^c}{\partial \boldsymbol{D} \partial \boldsymbol{\sigma}} : \left(\rho_0 \frac{\partial^2 \psi^c}{\partial \boldsymbol{\sigma} \partial \boldsymbol{\sigma}} \right)^{-1} : \rho_0 \frac{\partial^2 \psi^c}{\partial \boldsymbol{\sigma} \partial \boldsymbol{D}} : \frac{\partial f}{\partial \boldsymbol{Y}} - \frac{\partial K}{\partial \kappa} \\ - \frac{\partial f}{\partial \boldsymbol{\sigma}} : \left(\rho_0 \frac{\partial^2 \psi^c}{\partial \boldsymbol{\sigma} \partial \boldsymbol{\sigma}} \right)^{-1} : \rho_0 \frac{\partial^2 \psi^c}{\partial \boldsymbol{\sigma} \partial \boldsymbol{D}} : \frac{\partial f}{\partial \boldsymbol{Y}}. \end{aligned} \tag{4.21}$$

The algorithmic tangential elasticity matrix $\boldsymbol{C}_{ATS}$ has expression

$$\boldsymbol{C}_{ATS} = \left(\rho_0 \frac{\partial^2 \psi^c}{\partial \boldsymbol{\sigma} \partial \boldsymbol{\sigma}} + \frac{1}{H} \rho_0 \frac{\partial^2 \psi^c}{\partial \boldsymbol{\sigma} \partial \boldsymbol{D}} : \frac{\partial f}{\partial \boldsymbol{Y}} \otimes \left[\frac{\partial f}{\partial \boldsymbol{\sigma}} + \frac{\partial f}{\partial \boldsymbol{Y}} : \rho_0 \frac{\partial^2 \psi^c}{\partial \boldsymbol{D} \partial \boldsymbol{\sigma}} \right] \right)^{-1}, \tag{4.22}$$

where

$$H = -\frac{\partial K}{\partial \kappa}. \tag{4.23}$$

Equations (4.21) – (4.23) are derived in appendix A.

In the program code the elastic compliance matrix $\boldsymbol{L}$ is assembled by the sub-subroutine *CM_ELASTIC*, the algorithmic tangential compliance matrix $\boldsymbol{L}_{ATS}$ is assembled by *CM_ATS* and derivative $f'(\lambda)$ is assembled by *JACO*. Partial derivatives in above expressions are supplied by sub-subroutines *DERY, DERSIG, SIGDAM* and *SIGSIG*. Matrix inversions and linear equation system solutions are handled by subroutines *INVERT, PFACT* and *SUBST* that are based on the book by Conte and de Boor (1987).

Instead of the algorithmic tangential elasticity matrix $\boldsymbol{C}_{ATS}$ the secant elasticity matrix $\boldsymbol{C}$ could also be used, but then the *quadratic convergence rate* of Newton-Raphson iteration procedure is lost. However, due to non-associative flow rule, $\boldsymbol{C}_{ATS}$ is an unsymmetric matrix. It can be computationally more expensive because equation system solvers cannot utilize symmetry properties to speed up computation. Furthermore, in order to ensure *full Newton-Raphson scheme*, $\boldsymbol{C}_{ATS}$ must be evaluated using the stress $\boldsymbol{\sigma}$, damage $\boldsymbol{D}$ and hardening-softening state variable κ values at the end of iteration step as stated in equation (4.16). (Ottosen and Ristinmaa 2005, sections 17 and 18)

```
SUBROUTINE UMAT(STRESS, STATEV, DDSDDE, SSE, SPD, SCD,                  &
     RPL, DDSDDT, DRPLDE, DRPLDT,                                       &
     STRAN, DSTRAN, TIME, DTIME, TEMP, DTEMP, PREDEF, DPRED, CMNAME,    &
     NDI, NSHR, NTENS, NSTATEV, PROPS, NPROPS, COORDS, DROT, PNEWDT,    &
     CELENT, DFGRD0, DFGRD1, NOEL, NPT, LAYER, KSPT, KSTEP, KINC)
```

Program 4.1. *Elmer's UMAT-interface based on the Abaqus convention.*

The maximum number of allowed consistency condition iterations is determined by input parameter *Miter*. If step 3 has not converged before it runs out of iterations, solution algorithm tries to achieve convergence by splitting the strain increment $\Delta\boldsymbol{\varepsilon}$ into smaller increments. Initial strain increment can be split into $Miter/10$ substeps. This helps the global system as only the most difficult local material integrations are split into smaller increments instead of splitting the global step size of all integration points. The substepping is handled in the *UMAT* subroutine body.

4.2 Implementing Material Model into Elmer

Elmer's UMAT subroutine interface is based on the Abaqus-convention (*Abaqus, Theory Manual* 2014, section 1.1.41). The actual subroutine interface is shown in program 4.1. All parameters and features of the UMAT interface are not yet supported in Elmer. The most major limitation is the lack of nonlinear geometry support. Elmer user subroutines are written in *Fortran 90* programming language instead of Abaqus's *Fortran 77*.

Elmer's UMAT interface is fairly new and still under development. In fact it was released during this thesis project. At the time of writing this thesis only linear geometry implementation in static structural analysis is supported. It has not been documented yet so this description is based on the source code file *ElasticSolve.F90* which is available on Elmer web page (see *Overview of Elmer,* Råback and Malinen 2018). The basic global solution procedure is explained in section 3.5.

Table 4.1 presents the UMAT interface arguments that are passed between Elmer's global solution procedure and UMAT subroutine. Note that unsupported arguments in program 4.1 are not shown in the table. The developed material model subroutine utilizes only 13 of those arguments; *STRESS, STATEV, DDSDDE, STRAN, DSTRAN, NDI, NSHR, NTENS, NSTATEV, PROPS, NPROPS, NOEL* and *NPT*. In addition, *PNEWDT* value is returned even though Elmer does not yet utilize this value.

Some basic error checking is performed at the beginning of the subroutine. Since the model is developed for 3D situations only, parameters *NDI* and *NSHR* must contain value 3 and parameter *NTENS* must be 6. Number of material parameters *NPROPS* must be 14 and state variable array size *NSTATEV* must be 7. If any of these conditions is violated, UMAT is terminated with an error message. In case of substep divisions, a message is delivered just for information. If the subroutine fails to solve constitutive relations, *PNEWDT* value

less than 1.0 is returned along with an error message to inform the FE solver about the issue. Return value is the ratio of succesfully converged substeps to maximum allowed substeps. In case of substepping message or convergence error message, element number *NOEL* and integration point number *NPT* are also provided. *PNEWDT* value 1.0 is returned if UMAT has solved constitutive relations succesfully.

Elmer postprocesses variables *STRESS* and *STATEV* supplied by UMAT subroutine. Stress field is generated from the *STRESS*-variable. It can be visualized in external visualization tool, for example in ParaView. Elmer can supply the component-wise stress fields, but principal stresses and -angles are not yet available for UMAT implementations. Principal values and -directions for damage tensor $\boldsymbol{D}$ are obtained as an eigenvalue solution. Due to limitations of ParaView software, damage tensor-related values are cell-wise averaged. This is not the best practice since huge differences can occur in damage values of individual integration points inside a single element.

4.3 Using the Material Model in Elmer

In order to use the material model in Elmer simulations, user needs to compile the *ElasticSolve.F90* solver module with the file that contains the material model subroutine. Instructions can be found in *Elmer Solver Manual* (Ruokolainen et al. 2018, section 18.5). Compiled library is then located in the *solver*-branch of Elmer's solver input file.

Once this has been done, the actual usage of the model is relatively straightforward for those used to work in Elmer environment. In addition to regular *ElasticSolve*-module options, table 4.2 presents UMAT-specific options that are placed in the solver input file. Table 4.3 explains material constants array that is located in *material*-branch of the solver input file.

Table 4.1. *User material subroutine interface arguments in Elmer*

Argument	**Math notation** (if described)	**Description**
Arguments to be updated in all situations		
STRESS	$\boldsymbol{\sigma}$	Elmer provides the stress vector at the beginning of time step. UMAT needs to update the stress according to the strain increment $\Delta\boldsymbol{\varepsilon}$.
STATEV	$\boldsymbol{D}, \kappa$	Elmer provides material state variables at the beginning of time step. UMAT is required to update state variables according to the strain increment $\Delta\boldsymbol{\varepsilon}$.
DDSDDE	$\frac{\partial\Delta\boldsymbol{\sigma}}{\partial\Delta\boldsymbol{\varepsilon}}$	UMAT supplies the material Jacobian matrix for the strain increment $\Delta\boldsymbol{\varepsilon}$.
SSE *, SPD *, SCD *	-	Elmer provides specific strain energy (SSE), plastic dissipation (SPD) and creep dissipation (SCD) at the beginning of time step. UMAT updates these. *Note: these are used for postprocessing purposes only.*
Parameters passed in for information		
STRAN	$\boldsymbol{\varepsilon}$	The strain vector at the beginning of time step.
DSTRAN	$\Delta\boldsymbol{\varepsilon}$	The strain increment from the beginning of time step to the current strain iterate.
TIME *	-	Pseudo-time value at the beginning of load step.
DTIME *	-	Pseudo-time increment of the load step.
TEMP *	-	Temperature at the beginning of load step.
CMNAME *	-	The material model name.
NDI	-	Number of direct stress components.
NSHR	-	Number of shear stress components.
NTENS	-	Stess and strain vector size.
NSTATEV	-	Number of material state variables.
PROPS	$E, \nu, A, B, \ldots$	Array of material constants.
NPROPS	-	Number of material constants.
NOEL	-	The element number.
NPT	-	The integration point number.

*Arguments not utilized by this specific UMAT subroutine.

Table 4.2. *User material model options in solver input file*

Command	Argument type	Description
Options that are required in solver-branch		
Use UMAT	Logical	Must be set to *true* in order to utilize UMAT subroutine.
Large Deflection	Logical	Must be set to *false* in order to utilize UMAT subroutine.
Optional options in solver-branch		
Initialize State Variables	Logical	If set to *true*, material state variables are initialized at the beginning of simulation by calling the UMAT subroutine with initial conditions and zero strain increment. Otherwise they are initialized as zeroes.
Output State Variables	Logical	If set to *true*, Elmer calculates the principal values and principal directions of the damage tensor $\boldsymbol{D}$.
Options that are required in material-branch		
Number of Material Constants	Integer	Must be set to 14.
Number of State Variables	Integer	Must be set to 7.

Table 4.3. *Structure of Material Constants-array in solver input file*

Array index	Material parameter	Description
(1)	E	Elastic modulus
(2)	ν	Poisson's ratio
(3)	A	Parameters related to initial damage surface
(4)	B	
(5)	k_1	
(6)	k_2	
(7)	σ_{c0}	Initial compressive strength
(8)	H_0	Hardening-softening parameters
(9)	κ_0	
(10)	a_1	
(11)	a_2	
(12)	α_1	Volume dilatation parameters
(13)	α_2	
(14)	*Miter*	Maximum number of consistency condition iterations

5. RESULTS

5.1 Simple Loading Cases

Three different simple loading cases are considered. Results are compared with the experimental findings of Kupfer et al. (1969). Experimental data contains results for various simple load cases. Tests have been carried out for similar concrete test specimens, so comparison between different loading cases is informative. Simulated loading cases are

1. Uniaxial compression test
2. Uniaxial tension test
3. Equibiaxial compression test.

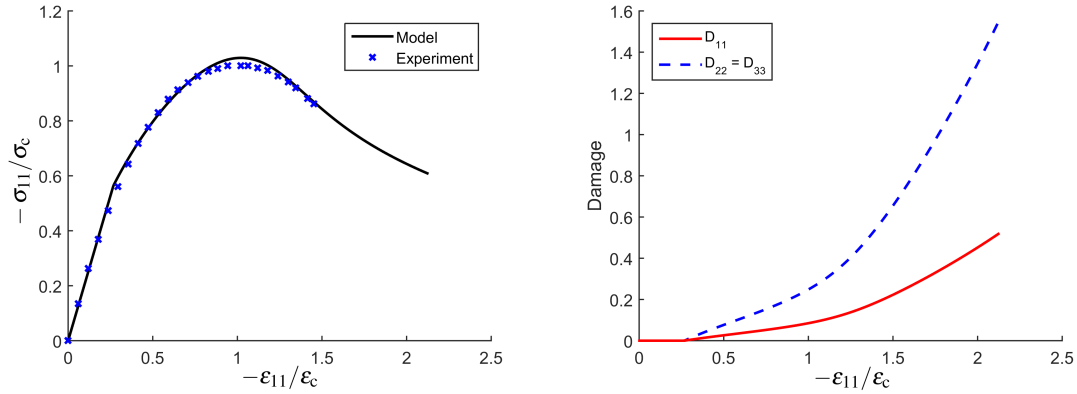
We have a change to evaluate the correctness of FE-implementation as the constitutive equations for these cases can be solved numerically by integrating single material point response. Then simulations are repeated using Elmer FE-implementation of the same material model. As the actual UMAT subroutine experiences similar stress states, we should see equal results. Also as the model uses same constitutive formulation as Hartikainen et al. (2018), we should be able to duplicate their results.

5.1.1 Uniaxial Compression

Because concrete can carry relatively high compressive loads, the uniaxial compression test is probably the most interesting simple loading case. Furthermore, concrete has a strong tendency to split axially in uniaxial compression. That makes this loading case even more interesting and also complicated in the sense of material modelling.

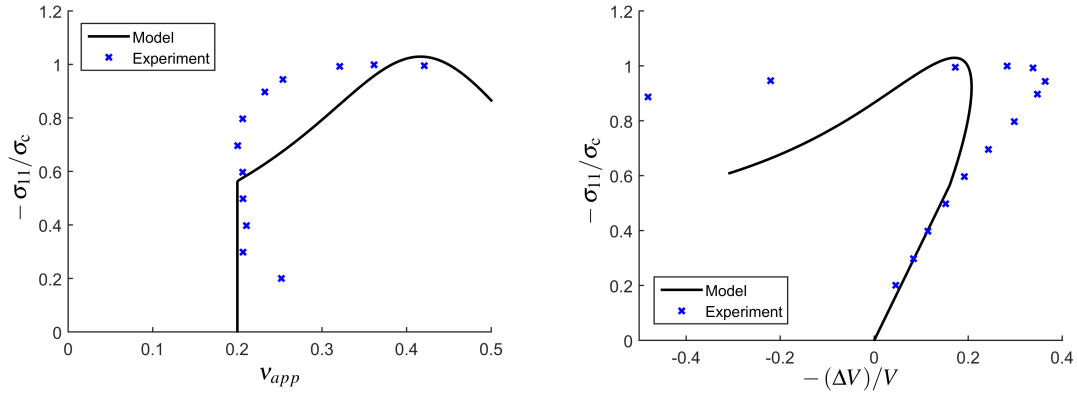
The simulated stress-strain response of the material is shown in figure 5.1a. Material parameters have beed adjusted using the initial elastic properties E and ν, initial limits of elastic behaviour and the ultimate uniaxial compression stress state $(\sigma_c, \varepsilon_c)$ in section 2.3.5. Thus the model predicts the ultimate uniaxial compression state well. The stress-strain response is also good between initial elastic limit $\sigma_{c0} = 0.549\,\sigma_c$ and ultimate stress state, as well as after the peak stress state. The load was applied only in 11-direction.

Figure 5.1b shows the damage evolution in different directions during the loading. Damaging is about 2.5 times more severe in transverse direction than in the loading direction in ultimate compressive stress state. The direction of damage represents normals to the cracking planes. This means that the model is capable of simulating the axial splitting phenomenon in compression.



(a) **(b)**

Figure 5.1. *Results for uniaxial compression test. Experimental results by Kupfer et al. (1969). (a) Stress-strain response (b) Damage evolution*



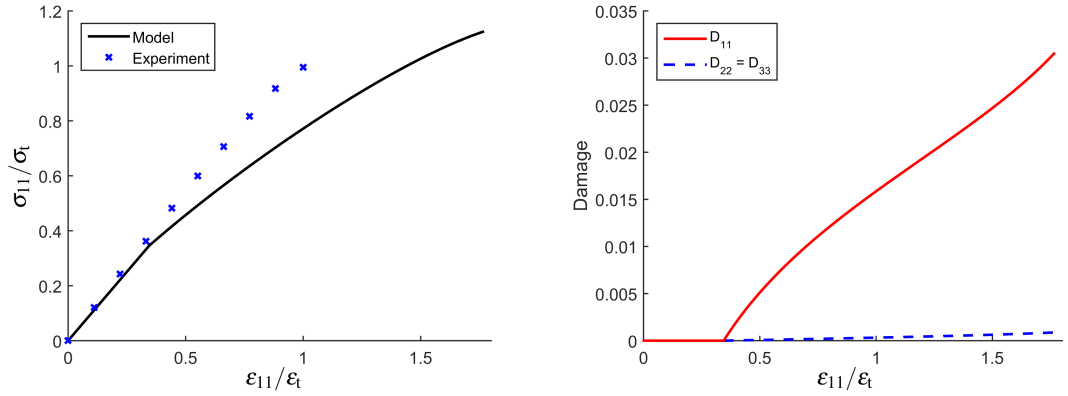
(a) **(b)**

Figure 5.2. *Results for uniaxial compression test. Experimental results by Kupfer et al. (1969). (a) Apparent Poisson's ratio (b) Volume dilatation*

The apparent Poisson's ratio ν_{app} and volume dilatation $\Delta V/V$ are shown in figure 5.2. When the ultimate stress is reached, apparent Poisson's ratio is 28 % higher and volume dilatation is 39 % lower compared to test data. This indicates that the model does not predict the transversal behaviour well.

5.1.2 Uniaxial Tension

The damage mechanism is different in tension than in compression. Therefore the ultimate tensile stress is much less than ultimate compressive strength. From the experimental data we can see that $\sigma_t = 0.09\,\sigma_c$ and $\varepsilon_t = 0.043\,\varepsilon_c$. Also, the cracking should now occur in transverse direction to the loading, indicating that damage should occur more in longitudinal direction.



(a) **(b)**

Figure 5.3. *Results for uniaxial tension test. (a) Stress-strain response (b) Damage evolution*

Figure 5.3 shows the simulated stress-strain and damage-strain responses in uniaxial tension. Simulated ultimate tensile stress is 12 % higher than the experimental data suggests and the ultimate tensile strain is 77 % too high. However, failure mode is correct as the damage developes almost entirely in the direction of loading.

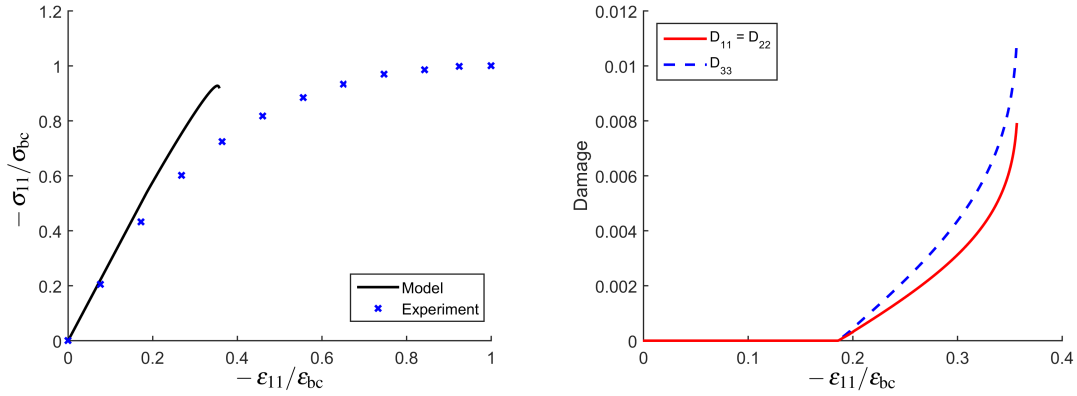
An important finding is that severe convergence problems occurred in the numerical calculations after tensile strain exceeded $1.77\,\varepsilon_t$. This behaviour seems realistic since the material cracks and experiences rapid loss of stiffness.

5.1.3 Biaxial Compression

The material was subjected to equibiaxial compression by applying the compressive stresses in 11 and 22-directions such that $\sigma_{11} = \sigma_{22}$. Experimental data suggests that $\sigma_{bc} = 1.16\,\sigma_c$ and $\varepsilon_{bc} = 1.23\,\varepsilon_c$. According to Kupfer et al. (1969) the micro-cracking in biaxial compression should occur mainly in the directions of loading. The macro-crack that ultimately caused the failure in testing was a bit more perpendicular to the loading directions than in uniaxial compression. That would suggest that the damaging should appear even more orientated to the transverse direction to the loading.

Figure 5.4a shows the simulated stress-strain response of the material. Simulated ultimate compressive stress is only 7.2 % lower than the experimental data suggests. But the stress-strain behaviour is almost linear and the ultimate compressive strain is 65 % too low. Similar to the uniaxial tension test, numerical convergence problems occurred after compressive stress exceeded $0.36\,\varepsilon_{bc}$.

The direction-wise damage values presented in figure 5.4b are closer to each others than in uniaxial compression case. This is opposite to the experimental findings. However, the perpendicular damage component is still larger than components in loading directions.



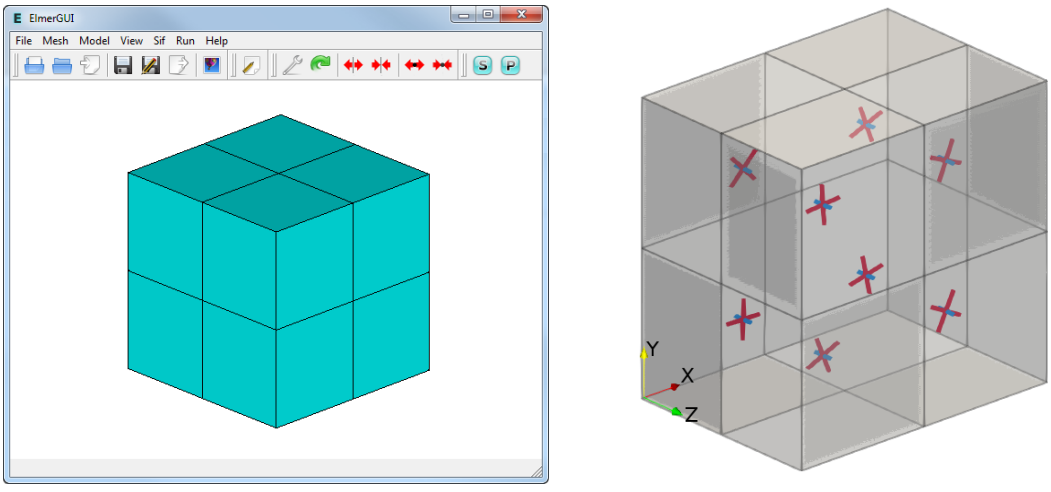
(a) **(b)**

Figure 5.4. *Results for equibiaxial compression test. (a) Stress-strain response (b) Damage evolution*

5.1.4 FEM Approach

The FE model was constructed in Elmer using eight trilinear (808) elements. The boundary conditions were applied such that the desired uniform stress states were achieved. Three simulations were run, one for each loading case. The FE-mesh is shown in figure 5.5a.

The simulated principal damage vectors are visualized as lines in figure 5.5b. They are orientated in the principal damage directions. The color and length of the line corresponds to the magnitude of the principal damage values. Damage situation is shown for ultimate uniaxial compression state. Red values correspond to damage value 0.25 and blue value is 0.084. Compression was applied in z-direction. Note that because of the plane-symmetric stress state, only the smallest principal damage direction is unique.



(a) **(b)**

Figure 5.5. *(a) FE calculation model in Elmer GUI (b) Principal damage values in exaggerately deformed ultimate uniaxial compression state.*

The FE solutions did not differ from those obtained by solving the constitutive equations directly. And as they were equal to the results obtained by Hartikainen et al., the FE implementation seems to be correct and we can proceed to simulate more general problems using FEM.

5.2 Compressed Pillar

The uniaxial compression test was repeated, but this time the lateral movement of compressed faces was prohibited. This way the stress state was no longer uniform due to Poisson's effect. As demonstrated by experiments (Peck 2012; Wu et al. 2018), this setup should produce an X-shaped failure region, with no damage near the centers of the compressed faces. Crack tips should point a bit outwards near the edges of the end faces. Simulated specimen was a prismatic pillar with dimensions (300 x 300 x 600) mm (width x depth x height). Material parameters were the same as in the previous cases.

The displacement-controlled loading was applied by displacing the nodes in the upper face downwards. Other than that, upper and lower face displacements were constrained. The simulation was run using 3 different mesh sizes of 2000 elements, 16000 elements (shown) and 31250 elements. Meshes were constructed using square-faced trilinear (808) elements. The computational model in its final state is shown in figure 5.6.

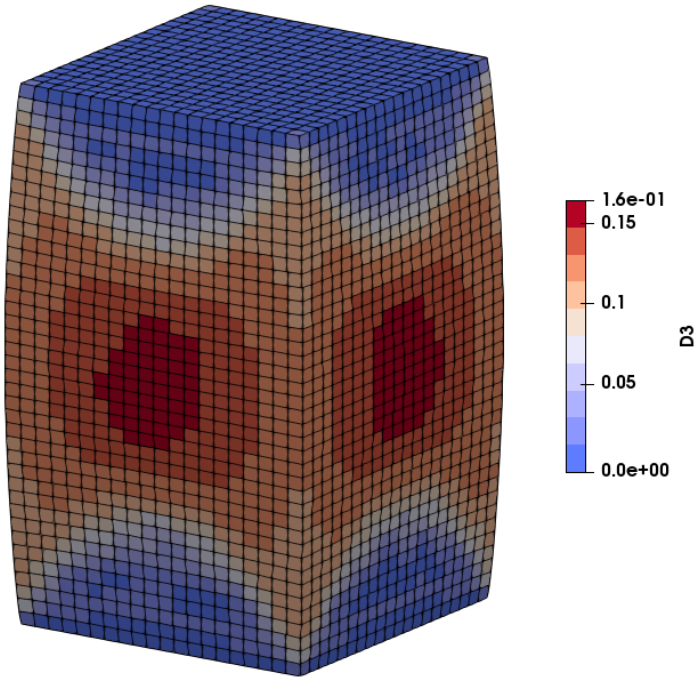
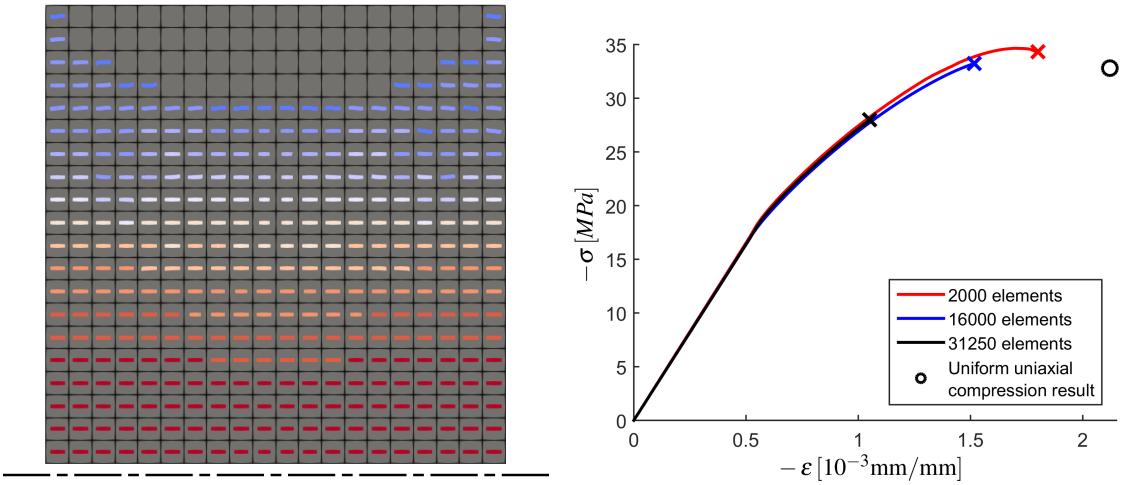


Figure 5.6. *Largest principal damage value in exaggerately displaced solution. Presented model has 16000 elements.*



(a) **(b)**

Figure 5.7. *(a) Principal directions associated with the largest principal damage values in the middle plane. Color coding is the same as in figure 5.6. Due to symmetry, only upper half is shown. (b) Engineering stress-strain plot for different mesh sizes. Crosses indicate points where converging stopped. Circle is peak-state from uniform uniaxial compression simulation.*

The visualized principal damage solution in figure 5.6 shows that the undamaged region corresponds to experimental findings. Also, the most damaged region is near the pillar's centroid. Figure 5.7a shows the pargest principal damage values and directions in the vertical mid-plane. From that we can see the direction change of the principal damage. Note again that the actual crack planes are perpendicular to the damage directions shown.

Engineering stress-strain relations for different mesh sizes are shown in figure 5.7b. The mesh size had little effect on the obtained stress-strain relation. But when mesh size was increased, the strain localization started earlier and caused the solution stopped converging. In all cases the problems first occurred in integration points near the mid-height nodes of the pillar. Compared to the uniform uniaxial compression results, the ultimate compressive stress value was increased due to the stiffening effect of the confinement near the end faces. It also reduced the ultimate compressive strain.

5.3 Notched Beam

Lastly, a beam with a notch in the underside was simulated. Calculation model with boundary conditions is illustrated in figure 5.8. Beam was loaded by forces $0.13P$ and P. This test setup causes mixed-mode fracturing near the notch. Main fracture modes are crack opening and crack sliding. Results are compared with experimental data by Arrea and Ingraffea (1982), cited by Rots et al. (1985). The width of the original experiment specimen was 156 mm, but only 10 mm wide section is modelled in plane strain and plane stress state. Simulations were run using models with 347 elements, 975 elements (shown) and 3072 elements. A total of 6 simulations were run. All models had 1 element over thickness.

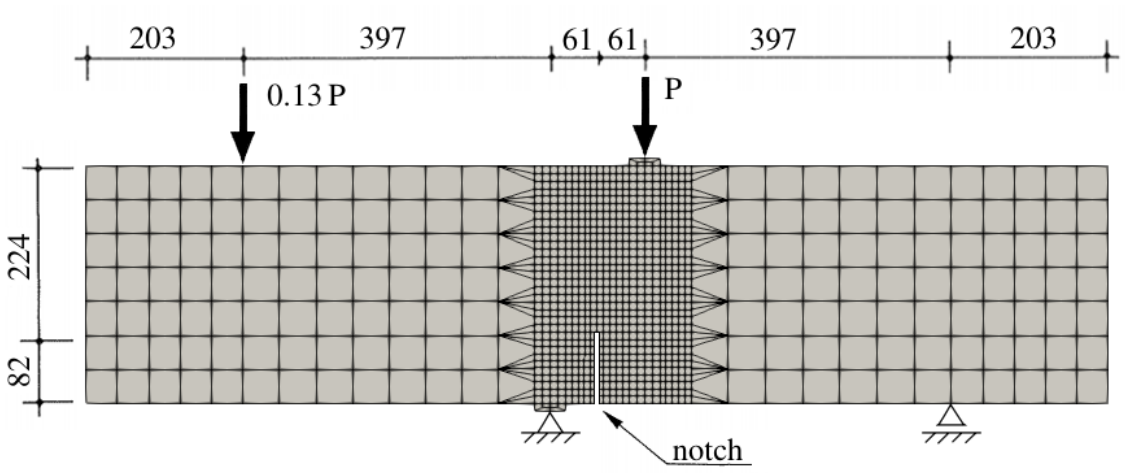


Figure 5.8. *FE mesh of the notched beam with boundary conditions. Dimensions in mm.*

Only the finer center part of the mesh was modelled as damageable concrete. Rest of the beam was linear elastic. Material parameters were the same as in the previous cases. Under the 2 centermost boundary conditions a (40 x 20) mm steel strip was modelled to distribute point loads to the concrete. Steel was linear elastic with elastic modulus 210 GPa and Poisson's ratio 0.3.

All simulations stopped converging at fairly early state. Compared to the experimental data, all models stopped converging at approximately half of the ultimate load regardless of the mesh size or whether plane strain or plane stress state was simulated. The stopping point is also the point where experimental data started showing nonlinear behaviour. Studying the unconverged simulation results reveals that the material softening always localized in one single material point. The problematic point is located near the crack tip.

However, the initial damage patterns illustrated in figures 5.9 and 5.10 is similar to the crack path visualized by Rots et al. (1985). Simulation results were similar in all 6 calculation models.

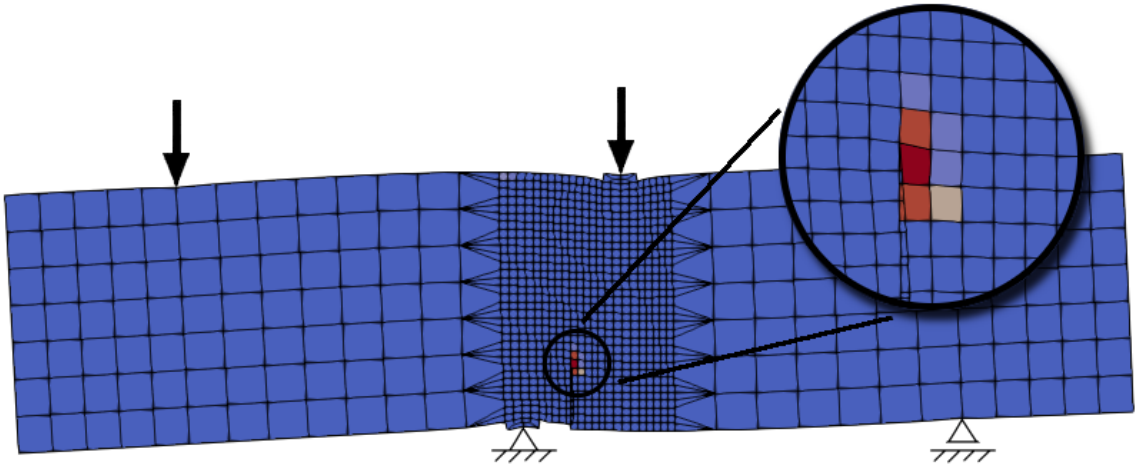
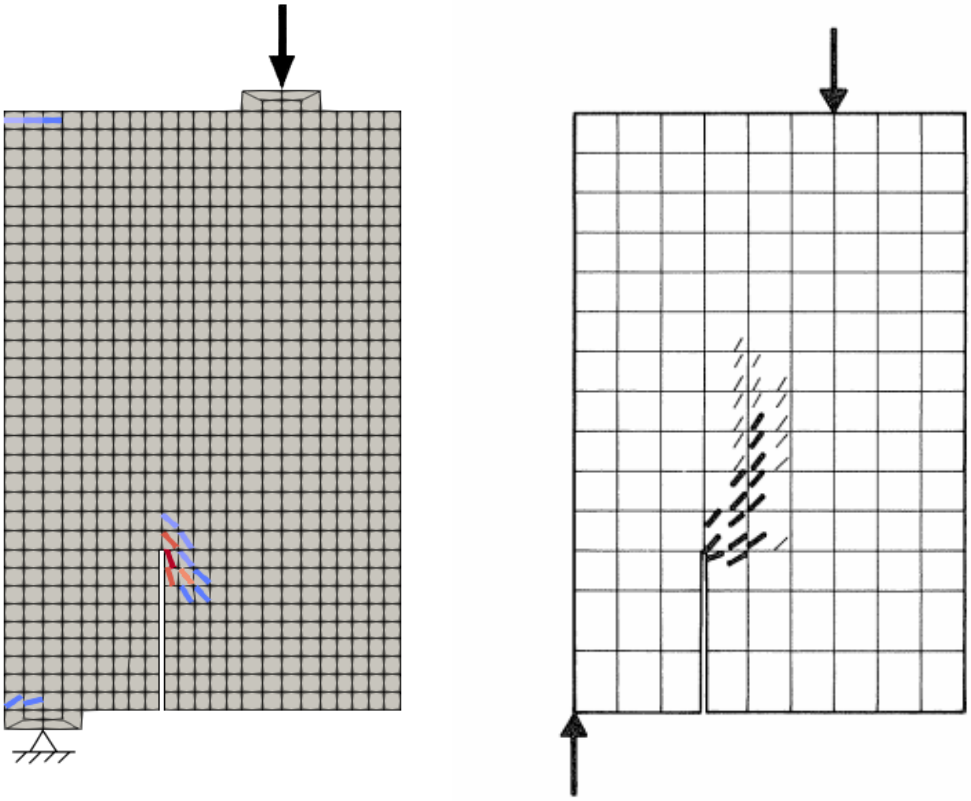


Figure 5.9. *Largest damage principal value in last converged solution. Deformations are exaggerated.*



(a) **(b)**

Figure 5.10. *(a) Principal directions associated with the largest principal damage value in the 975-element–model. (b) Crack path visualization by Rots et al. (1985).*

6. CONCLUSION

In this thesis an anisotropic continuum damage material model proposed by Yaghoubi et al. (2014) and Hartikainen et al. (2018) was formulated and implemented into existing FE-software. Thermodynamical basics leading to the material model were reviewed. Also the method for solving FE simulations containing nonlinear user-defined material models was shown as well as the detailed implementation of the specific material model as an independent UMAT subroutine.

The proposed material model simulated well the uniaxial compression of a concrete specimen. The stress-strain relation was accurate and it managed to generate the axial splitting failure mode. The performance under uniaxial tension and biaxial compression was not as good. Still, the model predicted correct failure modes.

Model was implemented into Elmer FE-software package through its UMAT-interface. Elmer was chosen as it is an open-source FE code featuring UMAT capabilities for structural analysis. As Elmer's UMAT-interface is designed to mimic Abaqus's convention for user-defined material models, the developed UMAT-subroutine could be relatively easily converted to Abaqus or other FE-solvers. The main modification would be to convert the subroutine from Fortran 90 language to Fortran 77 that Abaqus's user subroutines are written in.

Elmer FE-implementation of the material model was verified with the simplier loading cases. Then it was tested in some more general case simulations. Despite the convergence issues in numerical computations, the formation of the initial failure was predicted correctly. However, the ultimate failure state in more complex simulations was hard to reach. Still, the material model showed promising preliminary results. With further development it could become a very useful tool for modelling quasi-brittle materials beyond the elastic limit.

Elmer postprocesses some of the simulation results. It records the displacement field, strain field, stress field and calculates the principal values and principal directions of the damage tensor. However, the strain and stress fields are only output in global coordinate orientation. No principal values or any equivalent reference stresses are calculated. Also, the damage results are averaged cell-wise which is not ideal representation for damage variable.

As the model features non-associative flow rule and rate-independent strain hardening-softening, the boundary problem is ill-posed in the softening region and causes problems in FEM simulations. It is well known that this kind of material models show mesh-dependent FE solutions. However, even though all the FE simulations shown in chapter

5 were run using three different computation meshes, only few of them converged at all. Problems occurred after the first material point reached its ultimate stress state. The mesh-dependency did not show up in the results other in the form of convergence rate. Therefore mesh-dependency of the final solutions cannot yet be verified qualitatively.

These problems could be overcome by strain-rate regularization or element-size–dependent hardening-softening behaviour. By taking higher order damage tensor terms into account or adapting non-local constitutive relations could also help.

Other recommended future revisions would be to reformulate the specific potential functions that govern the material behaviour. The current model is formulated using the complementary part of Helmholtz free energy, which results a force-based formulation. Since FEM is a displacement-based method, the implementation of the model requires a lot of matrix inversions. In its current form the force-based formulation has no advantages. Therefore the model would be significantly more efficient if formulated using the actual Helmholtz free energy and related dissipation potential. Also some modification to the potential functions could be done in order to improve material response under more general loading cases — the biaxial compression for example.

Since discontinuous Galerkin elements have been proven to work well in fracture problems, they could be introduced into this model. Elmer already supports them. In order to simulate the cyclic behaviour, the model should include anisotropic hardening-softening. Also plastic strains could be taken into account, although they are speculated to be negligible.

In order to produce a useful tool for engineering, a systematic method for defining the material parameters should be developed. Preferably the parameters should be determined automatically according to user-inputted experimental failure stresses. A readily-available material library for most common materials could also be developed.

REFERENCES

Abaqus, Theory Manual (2014). Version 6.14.

Arrea, M. and Ingraffea, A. (1982), 'Mixed mode crack propagation in mortar and concrete', *Report 81-13* .

Basista, M. (2003), Micromechanics of damage in brittle solids, *in* J. Skrzypek and A. Ganczarski, eds, 'Anisotropic Behaviour of Damaged Materials', Vol. 9 of *Lecture Notes in Applied and Computational Mechanics*, Springer-Verlag, pp. 221–258.

Coleman, B. D. and Noll, W. (1963), 'The thermodynamics of elastic materials with heat conduction and viscosity', *Archive for Rational Mechanics and Analysis* **13**, 167–178.

Conte, S. and de Boor, C. (1987), *Elementary numerical analysis - An algorithmic approach*, McGraw-Hill, US.

Coulomb, C. A. (1776), 'Essai sur une application des regles de maximis & minimis a quelques problemes de statique: relatifs a l'architecture', *Memoires de mathematique et physique presente a l'Academie royale des sciences* **7**, 343–382. (in French).

Dragon, A., Halm, D. and Désoyer, T. (2000), 'Anisotropic damage in quasi-brittle solids: modelling, computational issues and applications', *Computer Methods in Applied Mechanics and Engineering* **183**(3-4), 331–352.

Drucker, D. C. and Prager, W. (1952), 'Soil mechanics and plastic analysis for limit design', *Quarterly of Applied Mathematics* **10**(2), 157–165.

Edelen, D. G. (1972), 'A nonlinear onsager theory of irreversibility', *International Journal of Engineering Science* **10**, 481–490.

Eringen, A. C. (1975), Thermodynamics of continua, *in* 'Continuum Physics', Vol. 2, Academic Press, New York, USA, pp. 89–127.

Frémond, M. (2002), *Non-Smooth Thermomechanics*, Springer, Berlin.

Hartikainen, J., Kolari, K., Kouhia, R. and Vilppo, J. (2018), Anisotropic damage model for concrete. Unpublished Manuscript.

Kachanov, L. (1958), 'Time of the rupture process under creep conditions', *Izdatielistvo Nauka, Moscow* (8), 26–31. (in Russian).

Kachanov, L. (1974), 'Foundations of fracture mechanics', *Izdatielistvo Nauka, Moscow* . (in Russian).

Kupfer, H., Hilsdorf, H. and Rüsch, H. (1969), 'Behaviour of concrete under biaxial stresses', *Journal of the American Concrete Institute* **66**(8), 656–666.

Lubliner, J., Oliver, J., Oller, S. and Oñate, E. (1989), 'A plastic-damage model for concrete', *International Journal of Solids and Structures* **25**(3), 299–326.

Malvern, L. E. (1969), *Introduction to the Mechanics of a Continuous Medium*, Prentice-Hall, New Jersey, USA.

Mohr, O. (1900), 'Welche umstände bedingen die elastizitätsgrenze und den bruch eines materials?', *Zeit des Ver Deut Ing* **44**, 1524–1530. (in German).

Murakami, S. (1988), 'Mechanical modeling of material damage', *Journal of Applied Mechanics* **55**, 280–286.

Murakami, S. (2012), *Continuum Damage Mechanics*, Springer Verlag, DE.

Onsager, L. (1931*a*), 'Reciprocal relations in irreversible processes. i.', *Physical Review* **37**, 405–426.

Onsager, L. (1931*b*), 'Reciprocal relations in irreversible processes. ii.', *Physical Review* **38**, 2265–2279.

Ottosen, N. S. (1977), 'A failure criterion for concrete', *Journal of the Engineering Mechanics, ASCE* **103**(EM4), 527–535.

Ottosen, N. S. and Ristinmaa, M. (2005), *The Mechanics of Constitutive Modeling*, Elsevier Science, UK.

Peck, M. (2012), *Concrete: Design, Construction, Examples*, Walter de Gruyter GmbH, DE.

Råback, P. and Malinen, M. (2018), *Overview of Elmer*, CSC – IT Center for Science.
URL: *http://www.csc.fi/elmer*

Råback, P., Malinen, M., Ruokolainen, J., Pursula, A. and Zwinger, T. (2018), *Elmer Models Manual*, CSC – IT Center for Science.
URL: *http://www.csc.fi/elmer*

Rots, J., Nauta, P., Kuster, G. and Blaauwendraad, J. (1985), 'Smeared crack approach and fracture localization in concrete', *HERON* **30**(1).

Ruokolainen, J., Malinen, M., Råback, P., Zwinger, T., Pursula, A. and Byckling, M. (2018), *ElmerSolver Manual*, CSC – IT Center for Science.
URL: *http://www.csc.fi/elmer*

Tadmor, E. B., Miller, R. E. and Elliott, R. S. (2012), *Continuum mechanics and thermodynamics : from fundamental concepts to governing equations*, Cambridge University Press, Cambridge, UK.

Wriggers, P. (2008), *Nonlinear Finite Element Methods*, Springer-Verlag Berlin Heidelberg, DE.

Wu, B., Yu, Y. and Chen, Z. (2018), 'Compressive behaviors of prisms made of demolished concrete lumps and fresh concrete', *Applied Sciences* **8**, 743.

Yaghoubi, S. T., Kouhia, R., Hartikainen, J. and Kolari, K. (2014), 'A continuum damage model based on ottosen's four parameter failure criterion for concrete', *Rakenteiden Mekaniikka (Journal of Structural Mechanics)* **47**, 50–60.

APPENDIX A: MATHEMATICAL DERIVATIONS

A.1 Second Order Partial Derivatives of Free Energy Function

Second order derivatives of free energy are needed for the later expressions. Their expressions are 4th order tensors. They can be derived by differentiating constitutive relation (2.70) as

$$\begin{aligned}\rho_0 \frac{\partial^2 \psi^{\mathrm{c}}}{\partial \boldsymbol{\sigma} \partial \boldsymbol{\sigma}} = \frac{\partial \boldsymbol{\varepsilon}}{\partial \boldsymbol{\sigma}} &= \frac{1+\nu}{E}\frac{\partial \boldsymbol{\sigma}}{\partial \boldsymbol{\sigma}} - \frac{\nu}{E}(1+\alpha_1 \operatorname{tr}\boldsymbol{D})\frac{\partial (\operatorname{tr}\boldsymbol{\sigma})}{\partial \boldsymbol{\sigma}}\boldsymbol{I} + \frac{\alpha_2}{E}\left(\frac{\partial \boldsymbol{\sigma}}{\partial \boldsymbol{\sigma}}\boldsymbol{D} + \boldsymbol{D}\frac{\partial \boldsymbol{\sigma}}{\partial \boldsymbol{\sigma}}\right) \\ &= \frac{1+\nu}{E}\boldsymbol{II} - \frac{\nu}{E}(1+\alpha_1 \operatorname{tr}\boldsymbol{D})\boldsymbol{I}\otimes\boldsymbol{I} + \frac{\alpha_2}{E}\left(\boldsymbol{I}\otimes\boldsymbol{D} + \boldsymbol{D}\otimes\boldsymbol{I}\right),\end{aligned} \tag{A.1}$$

where $\boldsymbol{II} = \delta_{ik}\delta_{jl}$ is the 4th order identity tensor. Symbols δ_{ik} and δ_{jl} are *Kronecker deltas*. Similarly we can obtain

$$\begin{aligned}\rho_0 \frac{\partial^2 \psi^{\mathrm{c}}}{\partial \boldsymbol{D} \partial \boldsymbol{\sigma}} = \frac{\partial \boldsymbol{\varepsilon}}{\partial \boldsymbol{D}} &= \frac{1+\nu}{E}\frac{\partial \boldsymbol{\sigma}}{\partial \boldsymbol{D}} - \frac{\nu}{E}\left(1+\alpha_1 \frac{\partial (\operatorname{tr}\boldsymbol{D})}{\partial \boldsymbol{D}}\right)(\operatorname{tr}\boldsymbol{\sigma})\boldsymbol{I} + \frac{\alpha_2}{E}\left(\boldsymbol{\sigma}\frac{\partial \boldsymbol{D}}{\partial \boldsymbol{D}} + \frac{\partial \boldsymbol{D}}{\partial \boldsymbol{D}}\boldsymbol{\sigma}\right) \\ &= -\frac{\nu}{E}(1+\alpha_1 \operatorname{tr}\boldsymbol{\sigma})\boldsymbol{I}\otimes\boldsymbol{I} + \frac{\alpha_2}{E}\left(\boldsymbol{\sigma}\otimes\boldsymbol{I} + \boldsymbol{I}\otimes\boldsymbol{\sigma}\right).\end{aligned} \tag{A.2}$$

As can be seen, expression (A.2) is symmetric and thus

$$\rho_0 \frac{\partial^2 \psi^{\mathrm{c}}}{\partial \boldsymbol{D} \partial \boldsymbol{\sigma}} = \rho_0 \frac{\partial^2 \psi^{\mathrm{c}}}{\partial \boldsymbol{\sigma} \partial \boldsymbol{D}}. \tag{A.3}$$

By utilizing relation (2.73) we can derive expression

$$\rho_0 \frac{\partial^2 \psi^{\mathrm{c}}}{\partial \kappa^2} = -\frac{\partial K}{\partial \kappa} = -\frac{\partial}{\partial \kappa}\left[H_0 \frac{a_1 (\kappa/\kappa_0)^2 + (\kappa/\kappa_0)}{a_2 (\kappa/\kappa_0)^2 + 1}\right] = H_0 \kappa_0 \frac{a_2\kappa^2 - 2a_1\kappa_0\kappa - \kappa_0^2}{(a_2\kappa^2 + \kappa_0^2)^2}. \tag{A.4}$$

A.2 Subgradients of the Dissipation Potential Function

Subgradients of the reformulated damage surface (2.64) are needed in order to solve the evolution of the internal varibles $\boldsymbol{D}$ and κ, to solve the consistency condition and to form the algorithmic tangential material Jacobian. The damage surface is

$$f(\boldsymbol{Y}, K; \boldsymbol{\sigma}) = \frac{A}{\sigma_{\mathrm{c0}}}\tilde{J}_2 + \Lambda\sqrt{\tilde{J}_2} + BI_1 - (\sigma_{\mathrm{c0}} + K). \tag{A.5}$$

Differentials with respect to $\boldsymbol{\sigma}$, $\boldsymbol{Y}$ and K are obtained via the chain rule as

$$\frac{\partial f}{\partial \boldsymbol{\sigma}} = \left(\frac{A}{\sigma_{c0}} + \sqrt{\tilde{J}_2} \frac{\partial \Lambda}{\partial \tilde{J}_2} + \frac{\Lambda}{2\sqrt{\tilde{J}_2}} \right) \frac{\partial \tilde{J}_2}{\partial \boldsymbol{\sigma}} + \sqrt{\tilde{J}_2} \frac{\partial \Lambda}{\partial \tilde{J}_3} \frac{\partial \tilde{J}_3}{\partial \boldsymbol{\sigma}} + \frac{\partial I_1}{\partial \boldsymbol{\sigma}}, \tag{A.6}$$

$$\frac{\partial f}{\partial \boldsymbol{Y}} = \left(\frac{A}{\sigma_{c0}} + \sqrt{\tilde{J}_2} \frac{\partial \Lambda}{\partial \tilde{J}_2} + \frac{\Lambda}{2\sqrt{\tilde{J}_2}} \right) \frac{\partial \tilde{J}_2}{\partial \boldsymbol{Y}} + \sqrt{\tilde{J}_2} \frac{\partial \Lambda}{\partial \tilde{J}_3} \frac{\partial \tilde{J}_3}{\partial \boldsymbol{Y}}, \tag{A.7}$$

$$\frac{\partial f}{\partial K} = -1. \tag{A.8}$$

Subdifferentials in the above expressions are

$$\frac{\partial \Lambda}{\partial \tilde{J}_2} = \begin{cases} -k_1 \cdot k_2 \dfrac{3\sqrt{3}}{4} \dfrac{\tilde{J}_3}{\tilde{J}_2^{5/2}} \dfrac{\sin\left[\frac{1}{3}\arccos(k_2 \cos 3\theta)\right]}{\sqrt{1-(k_2\cos 3\theta)^2}} & \text{if} \quad \cos 3\theta \geq 0 \\ -k_1 \cdot k_2 \dfrac{3\sqrt{3}}{4} \dfrac{\tilde{J}_3}{\tilde{J}_2^{5/2}} \dfrac{\sin\left[\frac{1}{3}\pi - \frac{1}{3}\arccos(-k_2 \cos 3\theta)\right]}{\sqrt{1-(k_2\cos 3\theta)^2}} & \text{if} \quad \cos 3\theta \leq 0 \end{cases}, \tag{A.9}$$

$$\frac{\partial \Lambda}{\partial \tilde{J}_3} = \begin{cases} k_1 \cdot k_2 \dfrac{\sqrt{3}}{2} \dfrac{1}{\tilde{J}_2^{3/2}} \dfrac{\sin\left[\frac{1}{3}\arccos(k_2 \cos 3\theta)\right]}{\sqrt{1-(k_2\cos 3\theta)^2}} & \text{if} \quad \cos 3\theta \geq 0 \\ k_1 \cdot k_2 \dfrac{\sqrt{3}}{2} \dfrac{1}{\tilde{J}_2^{3/2}} \dfrac{\sin\left[\frac{1}{3}\pi - \frac{1}{3}\arccos(-k_2 \cos 3\theta)\right]}{\sqrt{1-(k_2\cos 3\theta)^2}} & \text{if} \quad \cos 3\theta \leq 0 \end{cases}, \tag{A.10}$$

$$\frac{\partial \tilde{J}_2}{\partial \boldsymbol{\sigma}} = \left(\frac{3\alpha_1 \nu}{2\alpha_2} - \frac{1}{3} \right) (\mathrm{tr}\,\boldsymbol{\sigma}) \boldsymbol{I}, \tag{A.11}$$

$$\frac{\partial \tilde{J}_3}{\partial \boldsymbol{\sigma}} = \frac{E}{3\alpha_2} [\boldsymbol{Y} - (\mathrm{tr}\,\boldsymbol{Y}) \boldsymbol{I}] + \left(\frac{2}{9} - \frac{\alpha_1 \nu}{\alpha_2} \right) (\mathrm{tr}\,\boldsymbol{\sigma})^2 \boldsymbol{I}, \tag{A.12}$$

$$\frac{\partial \tilde{J}_2}{\partial \boldsymbol{Y}} = \frac{E}{2\alpha_2} \boldsymbol{I}, \tag{A.13}$$

$$\frac{\partial \tilde{J}_3}{\partial \boldsymbol{Y}} = \frac{E}{3\alpha_2} [\boldsymbol{\sigma} - (\mathrm{tr}\,\boldsymbol{\sigma}) \boldsymbol{I}], \tag{A.14}$$

$$\frac{\partial I_1}{\partial \boldsymbol{\sigma}} = \boldsymbol{I} \tag{A.15}$$

A.3 Jacobian for Consistency Condition

The rate of the damage surface $\dot{f}$ is needed in order to solve the consistency condition relation. Hence we develop the expression by starting from the equation (2.51)

$$\dot{f} = \frac{\partial f}{\partial \boldsymbol{\sigma}} : \dot{\boldsymbol{\sigma}} + \frac{\partial f}{\partial \boldsymbol{Y}} : \dot{\boldsymbol{Y}} + \frac{\partial f}{\partial K} \dot{K} = 0. \tag{A.16}$$

As the specific free energy is defined as $\psi^c = \psi^c(\boldsymbol{\sigma}, \boldsymbol{D}, \kappa)$, the strain-rate can be derived from the expression (2.70):

$$\dot{\boldsymbol{\varepsilon}} = \rho_0 \frac{\partial^2 \psi^c}{\partial \boldsymbol{\sigma} \partial \boldsymbol{\sigma}} : \dot{\boldsymbol{\sigma}} + \rho_0 \frac{\partial^2 \psi^c}{\partial \boldsymbol{\sigma} \partial \boldsymbol{D}} : \dot{\boldsymbol{D}} + \rho_0 \frac{\partial^2 \psi^c}{\partial \boldsymbol{\sigma} \partial \kappa} \dot{\kappa}, \tag{A.17}$$

where the last term vanishes. We can solve it for $\dot{\boldsymbol{\sigma}}$:

$$\dot{\boldsymbol{\sigma}} = -\left(\rho_0 \frac{\partial^2 \psi^c}{\partial \boldsymbol{\sigma} \partial \boldsymbol{\sigma}}\right)^{-1} : \left(\rho_0 \frac{\partial^2 \psi^c}{\partial \boldsymbol{\sigma} \partial \boldsymbol{D}} : \dot{\boldsymbol{D}} - \dot{\boldsymbol{\varepsilon}}\right). \tag{A.18}$$

When solving the consistency condition, strain increment $\Delta\boldsymbol{\varepsilon}$ is fixed so $\dot{\boldsymbol{\varepsilon}} = \mathbf{0}$. Rates for $\boldsymbol{Y}$ and K can be derived from expressions (2.71) and (2.73):

$$\dot{\boldsymbol{Y}} = \rho_0 \frac{\partial^2 \psi^c}{\partial \boldsymbol{D} \partial \boldsymbol{\sigma}} : \dot{\boldsymbol{\sigma}} + \rho_0 \frac{\partial^2 \psi^c}{\partial \boldsymbol{D} \partial \boldsymbol{D}} : \dot{\boldsymbol{D}} = \rho_0 \frac{\partial^2 \psi^c}{\partial \boldsymbol{D} \partial \boldsymbol{\sigma}} : \dot{\boldsymbol{\sigma}}, \tag{A.19}$$

$$\dot{K} = -\rho_0 \frac{\partial^2 \psi^c}{\partial \kappa^2} \dot{\kappa} = -\frac{\partial K}{\partial \kappa} \dot{\lambda}, \tag{A.20}$$

where relation $\rho_0(\partial^2 \psi^c / \partial \boldsymbol{D} \partial \boldsymbol{D}) = \mathbf{0}$ has been accounted. Combining relations (A.16) and (A.18) – (A.20) leads to expression

$$\begin{aligned} \dot{f} = -\frac{\partial f}{\partial \boldsymbol{Y}} : \rho_0 \frac{\partial^2 \psi^c}{\partial \boldsymbol{D} \partial \boldsymbol{\sigma}} : \left(\rho_0 \frac{\partial^2 \psi^c}{\partial \boldsymbol{\sigma} \partial \boldsymbol{\sigma}}\right)^{-1} : \rho_0 \frac{\partial^2 \psi^c}{\partial \boldsymbol{\sigma} \partial \boldsymbol{D}} : \frac{\partial f}{\partial \boldsymbol{Y}} \dot{\lambda} - \frac{\partial K}{\partial \kappa} \dot{\lambda} \\ - \frac{\partial f}{\partial \boldsymbol{\sigma}} : \left(\rho_0 \frac{\partial^2 \psi^c}{\partial \boldsymbol{\sigma} \partial \boldsymbol{\sigma}}\right)^{-1} : \rho_0 \frac{\partial^2 \psi^c}{\partial \boldsymbol{\sigma} \partial \boldsymbol{D}} : \frac{\partial f}{\partial \boldsymbol{Y}} \dot{\lambda}, \end{aligned} \tag{A.21}$$

from which we can gather the expression for Jacobian

$$\begin{aligned} f'(\lambda) = -\frac{\partial f}{\partial \boldsymbol{Y}} : \rho_0 \frac{\partial^2 \psi^c}{\partial \boldsymbol{D} \partial \boldsymbol{\sigma}} : \left(\rho_0 \frac{\partial^2 \psi^c}{\partial \boldsymbol{\sigma} \partial \boldsymbol{\sigma}}\right)^{-1} : \rho_0 \frac{\partial^2 \psi^c}{\partial \boldsymbol{\sigma} \partial \boldsymbol{D}} : \frac{\partial f}{\partial \boldsymbol{Y}} - \frac{\partial K}{\partial \kappa} \\ - \frac{\partial f}{\partial \boldsymbol{\sigma}} : \left(\rho_0 \frac{\partial^2 \psi^c}{\partial \boldsymbol{\sigma} \partial \boldsymbol{\sigma}}\right)^{-1} : \rho_0 \frac{\partial^2 \psi^c}{\partial \boldsymbol{\sigma} \partial \boldsymbol{D}} : \frac{\partial f}{\partial \boldsymbol{Y}}. \end{aligned} \tag{A.22}$$

A.4 Algorithmic Tangential Material Jacobian Matrix

The algorithmic tangential material Jacobian matrix $\boldsymbol{C}_{ATS}$ is needed in order to ensure that the Newton-Raphson solution algorithm maintains its quadratic convergence property. Since the material model utilizes only elastic strains, the algorithmic tangential elastic matrix turned out to be the same as the tangential continuum elasticity matrix.

We start from the relations (2.70), (2.71) and (2.73):

$$\boldsymbol{\varepsilon} = \rho_0 \frac{\partial \psi^c}{\partial \boldsymbol{\sigma}}, \qquad \boldsymbol{Y} = \rho_0 \frac{\partial \psi^c}{\partial \boldsymbol{D}}, \qquad K = -\rho_0 \frac{\partial \psi^c}{\partial \kappa}. \tag{A.23}$$

As the specific free energy is defined as $\psi^c = \psi^c(\boldsymbol{\sigma}, \boldsymbol{D}, \kappa)$, the strain-rate becomes

$$\dot{\boldsymbol{\varepsilon}} = \rho_0 \frac{\partial^2 \psi^c}{\partial \boldsymbol{\sigma} \partial \boldsymbol{\sigma}} : \dot{\boldsymbol{\sigma}} + \rho_0 \frac{\partial^2 \psi^c}{\partial \boldsymbol{\sigma} \partial \boldsymbol{D}} : \dot{\boldsymbol{D}} + \rho_0 \frac{\partial^2 \psi^c}{\partial \boldsymbol{\sigma} \partial \kappa} \dot{\kappa}. \tag{A.24}$$

The last term vanishes due to the specific expression for ψ^c. Damage rate is obtained from the dissipation potential (2.72)

$$\dot{\boldsymbol{D}} = \dot{\lambda} \frac{\partial f}{\partial \boldsymbol{Y}}, \tag{A.25}$$

where $f = f(\boldsymbol{Y}, \kappa; \boldsymbol{\sigma})$ is the reformulated Ottosen's damage surface. Multiplier $\dot{\lambda}$ is obtained from the consistency condition (2.51) as

$$\dot{f} = \frac{\partial f}{\partial \boldsymbol{\sigma}} : \dot{\boldsymbol{\sigma}} + \frac{\partial f}{\partial \boldsymbol{Y}} : \dot{\boldsymbol{Y}} + \frac{\partial f}{\partial K}\dot{K} = 0. \tag{A.26}$$

Rates of $\boldsymbol{Y}$ and K are

$$\dot{\boldsymbol{Y}} = \rho_0 \frac{\partial^2 \psi^c}{\partial \boldsymbol{D} \partial \boldsymbol{\sigma}} : \dot{\boldsymbol{\sigma}}, \qquad \dot{K} = -\rho_0 \frac{\partial^2 \psi^c}{\partial \kappa^2}\dot{\kappa} = -\frac{\partial K}{\partial \kappa}\dot{\lambda}, \tag{A.27}$$

and thus multiplier $\dot{\lambda}$ can be solved by inserting previous expressions to (A.26):

$$\dot{\lambda} = \frac{1}{H}\left[\frac{\partial f}{\partial \boldsymbol{\sigma}} + \frac{\partial f}{\partial \boldsymbol{Y}} : \rho_0 \frac{\partial^2 \psi^c}{\partial \boldsymbol{D} \partial \boldsymbol{\sigma}}\right] : \dot{\boldsymbol{\sigma}}, \tag{A.28}$$

where H is

$$H = \frac{\partial f}{\partial K}\frac{\partial K}{\partial \kappa} = -\frac{\partial K}{\partial \kappa}. \tag{A.29}$$

Substitution to equation (A.24) yields

$$\dot{\boldsymbol{\varepsilon}} = \left(\rho_0 \frac{\partial^2 \psi^c}{\partial \boldsymbol{\sigma} \partial \boldsymbol{\sigma}} + \frac{1}{H}\rho_0 \frac{\partial^2 \psi^c}{\partial \boldsymbol{\sigma} \partial \boldsymbol{D}} : \frac{\partial f}{\partial \boldsymbol{Y}} \otimes \left[\frac{\partial f}{\partial \boldsymbol{\sigma}} + \frac{\partial f}{\partial \boldsymbol{Y}} : \rho_0 \frac{\partial^2 \psi^c}{\partial \boldsymbol{D} \partial \boldsymbol{\sigma}}\right]\right) : \dot{\boldsymbol{\sigma}}. \tag{A.30}$$

The expression inside brackets is the tangential compliance tensor. It is 4th order tensor relating second order stress and strain tensors. As the numerical scheme utilizes Voigt's notation, it needs to be converted into second order tensor $\boldsymbol{L}_{ATS}$ that relates the stress vector and strain vector. Then the algorithmic tangential material Jacobian matrix the obtained by inverting it:

$$\boldsymbol{C}_{ATS} = \boldsymbol{L}_{ATS}^{-1}. \tag{A.31}$$

Partial derivatives in expression (A.30) are shown in sections A.2 and A.1.

APPENDIX B: UMAT SUBROUTINE SOURCE CODE

```
1  !**********************************************************************
2  SUBROUTINE UMAT(STRESS, STATEV, DDSDDE, SSE, SPD, SCD,                &
3       RPL, DDSDDT, DRPLDE, DRPLDT,                                     &
4       STRAN, DSTRAN, TIME, DTIME, TEMP, DTEMP, PREDEF, DPRED, CMNAME,  &
5       NDI, NSHR, NTENS, NSTATEV, PROPS, NPROPS, COORDS, DROT, PNEWDT,  &
6       CELENT, DFGRD0, DFGRD1, NOEL, NPT, LAYER, KSPT, KSTEP, KINC)
7
8  IMPLICIT NONE
9  ! Input parameters
10 CHARACTER*8 :: CMNAME
11 REAL*8 :: STRESS, STATEV, DDSDDE, DDSDDT, DRPLDE, STRAN, DSTRAN,      &
12   PREDEF, DPRED, PROPS, COORDS, DROT, DFGRD0, DFGRD1, SSE, SPD, SCD,  &
13   RPL, DRPLDT, TIME, DTIME, TEMP, DTEMP, PNEWDT, CELENT
14 INTEGER :: NTENS, NSTATEV, NPROPS, NOEL, NPT, LAYER, KSPT, KSTEP, KINC,&
15   NDI, NSHR
16 DIMENSION STRESS(NTENS), STATEV(NSTATEV), DDSDDE(NTENS, NTENS),       &
17   DDSDDT(NTENS), DRPLDE(NTENS), STRAN(NTENS), DSTRAN(NTENS),          &
18   TIME(2), PREDEF(1), DPRED(1), PROPS(NPROPS), COORDS(3),             &
19   DROT(3,3), DFGRD0(3,3), DFGRD1(3,3)
20 ! Local variables
21 INTEGER :: ICONV,MaxSubsteps,NumOfSubsteps,CurrSubstep
22 REAL*8 :: E,nu,alpha1,alpha2,k1,k2,PI,coms,A,B,kappa0, &
23   MaxNRiters,H0,a1,a2,kappa
24 REAL*8, DIMENSION(6)   :: EPSVEC,SIGVEC
25 REAL*8, DIMENSION(3,3) :: EPS,EPSO,DEPS,SIG,SIGO,DAM
26 REAL*8, DIMENSION(6,6) :: CM
27 ! Variables for subroutine INVERT
28 INTEGER :: IOUT
29 REAL*8  :: FSMAL
30 REAL*8, DIMENSION(9)  :: X
31 INTEGER, DIMENSION(9) :: IPIV
32 !************************************
33 FSMAL = 1.0e-30_8
34 PI    = 3.14159265359_8
35
36 ! Check that UMAT is called properly
37 IF (NDI /= 3 .OR. NSHR /= 3 .OR. NTENS /= 6) &
38   STOP 'UMAT: ONLY 3D GEOMETRY IS SUPPORTED'
39 IF (NPROPS /= 14) &
40   STOP 'UMAT: 14 MATERIAL PARAMETERS MUST BE GIVEN'
41 IF (NSTATEV /= 7) &
42   STOP 'UMAT: NUMBER OF STATE VARIABLES MUST BE 7'
43
44 ! Read material properties
```

```
E          = PROPS(1)
nu         = PROPS(2)
A          = PROPS(3)
B          = PROPS(4)
k1         = PROPS(5)
k2         = PROPS(6)
coms       = PROPS(7)
H0         = PROPS(8)
kappa0     = PROPS(9)
a1         = PROPS(10)
a2         = PROPS(11)
alpha1     = PROPS(12)
alpha2     = PROPS(13)
MaxNRiters = PROPS(14)

! In case of convergence problems, the step is divided into substeps.
! Maximum number of substeps is defined by parameter MaxNRiters:
MaxSubsteps = CEILING(MaxNRiters/10._8) + 1

! A simple loop which first trys to get solution with 1 substep, then
! with 2 and so on. Maximum number of substeps is defined by parameter
! 'MaxSubsteps'. If convergence is reached, looping stops and updated
! values are returned. If solution diverges, looping starts all over
! again from initial strain value 'stran' but step size is reduced.
DO NumOfSubsteps = 1,MaxSubsteps

  ! Read initial stress from input parameter STRESS
  SIGO(1,1) = STRESS(1)
  SIGO(2,2) = STRESS(2)
  SIGO(3,3) = STRESS(3)
  SIGO(1,2) = STRESS(4)
  SIGO(1,3) = STRESS(5)
  SIGO(2,3) = STRESS(6)
  SIGO(2,1) = SIGO(1,2)
  SIGO(3,1) = SIGO(1,3)
  SIGO(3,2) = SIGO(2,3)

  ! Read initial strain from input parameter STRAN. STRAN contains
  ! engineering strain vector and EPSO is infinitesimal strain tensor.
  EPSO(1,1) = STRAN(1)
  EPSO(2,2) = STRAN(2)
  EPSO(3,3) = STRAN(3)
  EPSO(1,2) = STRAN(4)*0.5_8
  EPSO(1,3) = STRAN(5)*0.5_8
  EPSO(2,3) = STRAN(6)*0.5_8
  EPSO(2,1) = EPSO(1,2)
  EPSO(3,1) = EPSO(1,3)
  EPSO(3,2) = EPSO(2,3)

  ! Read strain increment from input parameter "DSTRAN"
  DEPS(1,1) = DSTRAN(1)
```

```
  DEPS(2,2) = DSTRAN(2)
  DEPS(3,3) = DSTRAN(3)
  DEPS(1,2) = DSTRAN(4)*0.5_8
  DEPS(1,3) = DSTRAN(5)*0.5_8
  DEPS(2,3) = DSTRAN(6)*0.5_8
  DEPS(2,1) = DEPS(1,2)
  DEPS(3,1) = DEPS(1,3)
  DEPS(3,2) = DEPS(2,3)

  ! Read initial state variables from input parameter "STATEV"
  DAM(1,1) = STATEV(1)
  DAM(2,2) = STATEV(2)
  DAM(3,3) = STATEV(3)
  DAM(1,2) = STATEV(4)
  DAM(1,3) = STATEV(5)
  DAM(2,3) = STATEV(6)
  DAM(2,1) = DAM(1,2)
  DAM(3,1) = DAM(1,3)
  DAM(3,2) = DAM(2,3)
  kappa    = STATEV(7)

  ! Reduce strain increment DEPS according to number of substeps:
  DEPS = DEPS/(DBLE(NumOfSubsteps))

  ! Start solving individual substeps from substep number 1 to substep
  ! number 'NumOfSubsteps'
  DO CurrSubstep = 1,NumOfSubsteps

    ! Run the actual calculation
    CALL UPDAT(E,nu,A,B,k1,k2,coms,H0,kappa0,a1,a2,alpha1,alpha2,PI, &
               FSMAL,MaxNRiters,SIG,EPSO,DEPS,DAM,kappa,ICONV,CM)

    ! Check if the solution converged
    IF(ICONV > 0) THEN                              ! Solution converged

      ! Update initial strain tensor for the next substep.
      ! SIG, DAM and kappa are already updated by routine 'UPDAT'
      EPSO = EPSO+DEPS

    ELSE                                         ! solution did not converge
      IF(NumOfSubsteps == MaxSubsteps) THEN
        ! We exit the inner loop to demand more substeps and try again
        EXIT
      END IF
    END IF
  END DO

  ! Solving substeps has ended. Check the result.
  ! If solution is OK, exit loop and proceed to update output arguments
  IF(ICONV > 0) THEN
    EXIT
```

```
147   END IF
148
149   ! If solution was not acceptable, divide substep (if possible)
150   ! and try again
151   WRITE(*,*) 'UMAT: SUBSTEP DIVIDED INSIDE UMAT SUBROUTINE'
152   WRITE(*,*) 'ELEMENT NO ', NOEL, ', INTEGRATION POINT', NPT
153 END DO
154
155 ! Check result
156 ! If solution is not acceptible, print error message and suggest new
157 ! timestep size according to the number of substeps used and the
158 ! severity of convergence issue.
159 IF(ICONV < 0) THEN
160   WRITE(*,*) 'UMAT: MAXIMUM NUMBER OF ITERATIONS EXCEEDED INSIDE UMAT'
161   WRITE(*,*) 'ELEMENT NO ', NOEL, ', INTEGRATION POINT', NPT
162   PNEWDT = DBLE(CurrSubstep)/(DBLE(MaxSubsteps+1))
163
164 ! If the solution is acceptable, update stress, state variables,
165 ! (algorithmic) material Jacobian matrix and return pnewdt = 1.0,
166 ! which states that the substep converged succesfully
167 ELSE
168   ! Update stress
169   STRESS(1) = SIG(1,1)
170   STRESS(2) = SIG(2,2)
171   STRESS(3) = SIG(3,3)
172   STRESS(4) = SIG(1,2)
173   STRESS(5) = SIG(1,3)
174   STRESS(6) = SIG(2,3)
175
176   ! Update state variables
177   STATEV(1) = DAM(1,1)
178   STATEV(2) = DAM(2,2)
179   STATEV(3) = DAM(3,3)
180   STATEV(4) = DAM(1,2)
181   STATEV(5) = DAM(1,3)
182   STATEV(6) = DAM(2,3)
183   STATEV(7) = kappa
184
185   ! Elasticity matrix DDSDDE is inverse of CM
186   CALL INVERT(CM,DDSDDE,IPIV,6,X,FSMAL,IOUT)
187
188   ! Then we need to convert it matrix back to engineering convention:
189   DDSDDE(:,4) = DDSDDE(:,4)*0.5_8
190   DDSDDE(:,5) = DDSDDE(:,5)*0.5_8
191   DDSDDE(:,6) = DDSDDE(:,6)*0.5_8
192
193   ! PNEWDT must be 1.0 to indicate that step converged succesfully
194   PNEWDT = 1.0_8
195 END IF
196 END SUBROUTINE UMAT
197 !***********************************************************************
```

```
198
199 !*********************************************************************
200 SUBROUTINE UPDAT(E,nu,A,B,k1,k2,coms,H0,kappa0,a1,a2,alpha1,alpha2,PI, &
201   FSMAL,MaxNRiters,SIG,EPSO,DEPS,DAM,kappa,ICONV,CM)
202 ! This subroutine updates stress SIG and internal variables DAM and
203 ! kappa and returns algorithmic compliance matrix CM.
204 ! Output parameter ICONV tells the state of solution:
205 ! -1: did not convergence
206 !  0: still iterating
207 !  1: converged
208 IMPLICIT NONE
209 INTEGER :: II,ICONV
210 REAL*8 :: E,nu,A,B,k1,k2,coms,H0,kappa0,a1,a2,alpha1,alpha2,PI,        &
211   MaxNRiters,kappa,TrEPS,TrSIG,TrY,TrSIG2,TrSIGY,J2,J3,COTET,GAM,f,Dlam
212 REAL*8, DIMENSION(6)   :: SIGVEC,EPSVEC
213 REAL*8, DIMENSION(3,3) :: IDE,SIG,EPS,EPSO,DEPS,DAM,SIGDEV,SIGDE2,SIGDE3
214 REAL*8, DIMENSION(6,6) :: CM
215 ! Variables for subroutines PFACT and SUBST
216 INTEGER :: IOUT,IFLAG
217 REAL*8  :: FSMAL
218 REAL*8, DIMENSION(6)  :: D
219 INTEGER, DIMENSION(9) :: IPIV
220 !************************************
221 IDE = 0._8
222 DO II = 1,3
223  IDE(II,II) = 1._8
224 END DO
225
226 ! Total strain at the end of step
227 EPS = EPSO+DEPS
228
229 ! (1) Calculate the elastic trial stress.
230 ! If the damage tensor DAM is zero, we can use easier formula:
231 IF (MAXVAL(DABS(DAM)) < 1e-12) THEN
232   TrEPS = EPS(1,1)+EPS(2,2)+EPS(3,3)
233   SIG   = (E/(1._8+nu))*EPS+(nu*E/((1._8+nu)*(1._8-2._8*nu)))*TrEPS*IDE
234
235 ! If DAM is not zero, we calculate stresses via Voigt notation:
236 ELSE
237   ! Strains in Voigt notation
238   EPSVEC(1) = EPS(1,1)
239   EPSVEC(2) = EPS(2,2)
240   EPSVEC(3) = EPS(3,3)
241   EPSVEC(4) = EPS(1,2)
242   EPSVEC(5) = EPS(1,3)
243   EPSVEC(6) = EPS(2,3)
244
245   ! Elastic compliance matrix for current damage tensor
246   CALL CM_ELASTIC(E,nu,alpha1,alpha2,DAM,CM)
247
248   ! Calculate new stresses by solving the matrix equation
```

```
249   CAll PFACT(CM,6,D,IPIV,IFLAG,FSMAL)
250   CALL SUBST(CM,SIGVEC,EPSVEC,IPIV,6)
251
252   ! Stresses from Voigt notation to stress tensor
253   SIG(1,1) = SIGVEC(1)
254   SIG(2,2) = SIGVEC(2)
255   SIG(3,3) = SIGVEC(3)
256   SIG(1,2) = SIGVEC(4)
257   SIG(1,3) = SIGVEC(5)
258   SIG(2,3) = SIGVEC(6)
259   SIG(2,1) = SIG(1,2)
260   SIG(3,1) = SIG(1,3)
261   SIG(3,2) = SIG(2,3)
262 END IF
263
264 ! (2) Check the damage condition
265 TrSIG   = SIG(1,1)+SIG(2,2)+SIG(3,3)
266 SIGDEV  = SIG-(TrSIG/3._8)*IDE
267 SIGDE2 = MATMUL(SIGDEV,SIGDEV)
268 SIGDE3 = MATMUL(SIGDE2,SIGDEV)
269
270 J2    = 0.5_8*(SIGDE2(1,1)+SIGDE2(2,2)+SIGDE2(3,3))
271 J3    = (1._8/3._8)*(SIGDE3(1,1)+SIGDE3(2,2)+SIGDE3(3,3))
272 COTET = 1.5_8*DSQRT(3._8)*J3*((J2)**(-1.5))
273
274 IF (COTET > 0._8) THEN
275   GAM = k1*DCOS((1._8/3._8)*DACOS(k2*COTET))
276 ELSE
277   GAM = k1*DCOS(PI/3._8-(1._8/3._8)*DACOS(-k2*COTET))
278 END IF
279
280 f = A*J2/coms+GAM*DSQRT(J2)+B*TrSIG-coms &
281     -H0*(a1*(kappa/kappa0)**2+kappa/kappa0)/(a2*(kappa/kappa0)**2+1)
282
283 ! Check if material responds elastically or inelastically
284 IF(f > 0._8)THEN                                    ! Inelastic response
285   ! (3) Solve nonlinear material response
286   CALL NEWT(E,nu,A,B,k1,k2,coms,H0,kappa0,a1,a2,alpha1,alpha2,PI, &
287             IDE,FSMAL,MaxNRiters,SIG,EPS,DAM,kappa,Dlam,ICONV)
288
289   ! Update CM. Use algorithmic tangential material Jacobian if
290   ! multiplier Dlam is positive. Otherwise secant material Jacobian.
291   IF (Dlam < FSMAL) THEN
292    CALL CM_ELASTIC(E,nu,alpha1,alpha2,DAM,CM)
293   ELSE
294     CALL CM_ATS(E,nu,A,B,k1,k2,coms,H0,kappa0,a1,a2,alpha1,alpha2,PI,  &
295                 IDE,SIG,DAM,kappa,CM)
296   END IF
297
298 ELSE                                                ! Elastic response
299   ! Update CM
```

```
  CALL CM_ELASTIC(E,nu,alpha1,alpha2,DAM,CM)
  ICONV = 1
END IF
END SUBROUTINE UPDAT
!**********************************************************************

!**********************************************************************
SUBROUTINE NEWT(E,nu,A,B,k1,k2,coms,H0,kappa0,a1,a2,alpha1,alpha2,PI,  &
  IDE,FSMAL,MaxNRiters,SIG,EPS,DAM,kappa,Dlam,ICONV)
! If material responds irreversibly, this subroutine calculates the
! evolution of internal variables and updates stresses
IMPLICIT NONE
INTEGER :: ICONV
REAL*8  :: E,nu,A,B,k1,k2,coms,H0,kappa0,a1,a2,alpha1,alpha2,PI,kappa, &
  Dlam,MaxNRiters,TrSIG,TrSIG2,TrY,TrSIGY,J2,J3,COTET,GAM,f,DDlam,     &
  DDkappa,JAC,fRESID,NRiter
REAL*8, DIMENSION(3,3) :: IDE,SIG,EPS,DAM,DDAM,SIG2,Y,SIGY,DfY
REAL*8, DIMENSION(6)   :: SIGVEC,EPSVEC
REAL*8, DIMENSION(6,6) :: CM
! Variables for subroutines PFACT and SUBST
INTEGER :: IOUT,IFLAG
REAL*8  :: FSMAL
REAL*8, DIMENSION(6)  :: D
INTEGER, DIMENSION(9) :: IPIV
!***********************************
! NRiter is counter for Newton-Raphson iterations, ICONV determines
! whether convergence has been reached or not.
NRiter = 0._8
ICONV = 0

Dlam = 0._8

! Newton-Raphson iteration loop
DO WHILE (ICONV == 0)
  NRiter = NRiter + 1._8

  ! Current value of damage surface f
  TrSIG  = SIG(1,1)+SIG(2,2)+SIG(3,3)
  SIG2   = MATMUL(SIG,SIG)
  TrSIG2 = SIG2(1,1)+SIG2(2,2)+SIG2(3,3)
  Y      = -alpha1*nu* (TrSIG**2)*IDE/(2._8*E)+alpha2*SIG2/E
  TrY    = Y(1,1)+Y(2,2)+Y(3,3)
  SIGY   = MATMUL(SIG,Y)
  TrSIGY = SIGY(1,1)+SIGY(2,2)+SIGY(3,3)

  J2    = 0.5_8*(E*TrY/alpha2+((1.5_8*alpha1*nu/alpha2)-(1._8/3._8)) &
          *(TrSIG**2))
  J3    = ((E/alpha2)*(TrSIGY-TrSIG*TrY)+((2._8/9._8) &
          -(alpha1*nu/alpha2))*(TrSIG)**3)/3._8
  COTET = 1.5_8*DSQRT(3._8)*J3*((J2)**(-1.5))

```

```
351   IF ( COTET > 0._8 ) THEN
352     GAM = k1*DCOS((1._8/3._8)*DACOS(k2*COTET))
353   ELSE
354     GAM = k1*DCOS(PI/3._8-(1._8/3._8)*DACOS(-k2*COTET))
355   END IF
356
357   f = A*J2/coms+GAM*DSQRT(J2)+B*TrSIG-coms &
358       -H0*(a1*(kappa/kappa0)**2+kappa/kappa0)/(a2*(kappa/kappa0)**2+1)
359
360   ! Calculate derivatives DfY and JAC
361   CALL JACO(E,nu,A,B,k1,k2,coms,H0,kappa0,a1,a2,alpha1,alpha2,PI, &
362             IDE,FSMAL,SIG,DAM,kappa,DfY,JAC)
363
364   ! Iterative change of variable \lambda
365   DDlam = -f/JAC
366   Dlam  = Dlam+DDlam
367
368   ! Iterative change of damage tensor
369   CALL DERY(E,nu,A,B,k1,k2,coms,alpha1,alpha2,PI,IDE,SIG,DfY)
370   DDAM = DDlam*DfY
371   DAM  = DAM+DDAM
372
373   ! Iterative change of hardening/softening variable kappa
374   DDkappa = DDlam
375   kappa   = kappa+DDkappa
376
377   ! Calculate new stresses
378   ! Strains in Voigt notation
379   EPSVEC(1) = EPS(1,1)
380   EPSVEC(2) = EPS(2,2)
381   EPSVEC(3) = EPS(3,3)
382   EPSVEC(4) = EPS(1,2)
383   EPSVEC(5) = EPS(1,3)
384   EPSVEC(6) = EPS(2,3)
385
386   ! construct compliance matrix CM
387   CALL CM_ELASTIC(E,nu,alpha1,alpha2,DAM,CM)
388
389   ! Calculate new stresses
390   CAll PFACT(CM,6,D,IPIV,IFLAG,FSMAL)
391   CALL SUBST(CM,SIGVEC,EPSVEC,IPIV,6)
392
393   ! Stresses from Voigt notation to stress tensor
394   SIG(1,1) = SIGVEC(1)
395   SIG(2,2) = SIGVEC(2)
396   SIG(3,3) = SIGVEC(3)
397   SIG(1,2) = SIGVEC(4)
398   SIG(1,3) = SIGVEC(5)
399   SIG(2,3) = SIGVEC(6)
400   SIG(2,1) = SIG(1,2)
401   SIG(3,1) = SIG(1,3)
```

```
402   SIG(3,2) = SIG(2,3)
403
404   ! Calculate residual for f
405   fRESID = DABS(f/coms)
406   ! Check if f is converged
407   IF( fRESID < 1.0E-8_8 ) THEN                                  ! Converged
408     ICONV = 1
409   ELSE IF( NRiter > MaxNRiters-0.5_8 ) THEN                ! Miter exceeded
410     ICONV = -1
411   END IF
412
413   ! If solution is not converged or diverged and maximum number of
414   ! iterations is not exceeded, ICONV remains 0 and iteration continues.
415 END DO
416 END SUBROUTINE NEWT
417 !**********************************************************************
418
419 !**********************************************************************
420 SUBROUTINE JACO(E,nu,A,B,k1,k2,coms,H0,kappa0,a1,a2,alpha1,alpha2,PI,  &
421   IDE,FSMAL,SIG,DAM,kappa,DfY,JAC)
422 ! Calculates the Jacobian f'(\lambda) for Newton-Raphson iteration.
423 ! Also DfY is calculated
424 IMPLICIT NONE
425 REAL*8 :: E,nu,A,B,k1,k2,coms,H0,kappa0,a1,a2,alpha1,alpha2,PI,kappa,JAC
426 REAL*8, DIMENSION(3,3) :: IDE,SIG,DAM,DfY,DfS
427 REAL*8, DIMENSION(9,9) :: DSID,DSISI,INVDSI,WORKTENS
428 REAL*8, DIMENSION(9)   :: VDfY,VDfS,LCV1,LCV2
429 ! Variables for subroutine INVERT
430 INTEGER :: IOUT
431 REAL*8  :: FSMAL
432 REAL*8, DIMENSION(9)  :: X
433 INTEGER, DIMENSION(9) :: IPIV
434 !***************************************
435 ! Obtain necessary derivatives
436 CALL DERY(E,nu,A,B,k1,k2,coms,alpha1,alpha2,PI,IDE,SIG,DfY)
437 CALL DERSIG(E,nu,A,B,k1,k2,coms,alpha1,alpha2,PI,IDE,SIG,DfS)
438 CALL VEC(DfY,VDfY)
439 CALL VEC(DfS,VDfS)
440 CALL SIGDAM(E,nu,alpha1,alpha2,SIG,DSID)
441 CALL SIGSIG(E,nu,alpha1,alpha2,DAM,DSISI)
442
443 ! Invert DSISI to get INVDSI
444 CALL INVERT(DSISI,INVDSI,IPIV,9,X,FSMAL,IOUT)
445
446 ! Calculate damage surface Jacobian.
447 WORKTENS = MATMUL(INVDSI,DSID)
448 LCV1     = MATMUL(WORKTENS,VDfY)
449 LCV2     = MATMUL(DSID,LCV1)
450 JAC      = -DOT_PRODUCT(VDfY,LCV2)-DOT_PRODUCT(VDfS,LCV1) &
451            -H0*kappa0*(-a2*kappa**2+2*a1*kappa*kappa0+kappa0**2) &
452            /(a2*kappa**2 + kappa0**2)**2
```

```
453 END SUBROUTINE JACO
454 !**********************************************************************
455
456 !**********************************************************************
457 SUBROUTINE CM_ELASTIC(E,nu,alpha1,alpha2,DAM,CM)
458 ! Returns elastic compliance tensor CM for current DAM value
459 IMPLICIT NONE
460 REAL*8 :: E,nu,alpha1,alpha2,eta1,eta2,eta3
461 REAL*8, DIMENSION(3,3) :: DAM
462 REAL*8, DIMENSION(6,6) :: CM
463 !***********************************
464 eta1 = (1._8+nu)/E
465 eta2 = (nu/E)*(1._8 + alpha1*(DAM(1,1)+DAM(2,2)+DAM(3,3)))
466 eta3 = alpha2/E
467
468 CM = 0._8
469 CM(1,1) = eta1-eta2+2._8*eta3*DAM(1,1)
470 CM(1,2) = -eta2
471 CM(2,1) = CM(1,2)
472 CM(1,3) = -eta2
473 CM(3,1) = CM(1,3)
474 CM(1,4) = 2._8*eta3*DAM(1,2)
475 CM(4,1) = CM(1,4)*0.5_8
476 CM(1,5) = 2._8*eta3*DAM(1,3)
477 CM(5,1) = CM(1,5)*0.5_8
478 CM(2,2) = eta1-eta2+2._8*eta3*DAM(2,2)
479 CM(2,3) = -eta2
480 CM(2,4) = 2._8*eta3*DAM(1,2)
481 CM(2,6) = 2._8*eta3*DAM(2,3)
482 CM(3,2) = CM(2,3)
483 CM(4,2) = CM(2,4)*0.5_8
484 CM(6,2) = CM(2,6)*0.5_8
485 CM(3,3) = eta1-eta2+2._8*eta3*DAM(3,3)
486 CM(3,5) = 2._8*eta3*DAM(1,3)
487 CM(3,6) = 2._8*eta3*DAM(2,3)
488 CM(5,3) = CM(3,5)*0.5_8
489 CM(6,3) = CM(3,6)*0.5_8
490 CM(4,4) = eta1+eta3*DAM(1,1)+eta3*DAM(2,2)
491 CM(4,5) = eta3*DAM(2,3)
492 CM(4,6) = eta3*DAM(1,3)
493 CM(5,4) = CM(4,5)
494 CM(6,4) = CM(4,6)
495 CM(5,5) = eta1+eta3*DAM(1,1)+eta3*DAM(3,3)
496 CM(5,6) = eta3*DAM(1,2)
497 CM(6,5) = CM(5,6)
498 CM(6,6) = eta1+eta3*DAM(2,2)+eta3*DAM(3,3)
499 END SUBROUTINE CM_ELASTIC
500 !**********************************************************************
501
502 !**********************************************************************
503 SUBROUTINE CM_ATS(E,nu,A,B,k1,k2,coms,H0,kappa0,a1,a2,alpha1,alpha2,PI,&
```

```
504     IDE,SIG,DAM,kappa,CM)
505   ! Returns algorithmic tangential compliance matrix CM for current state
506   IMPLICIT NONE
507   INTEGER II,JJ
508   REAL*8 :: E,nu,A,B,k1,k2,coms,H0,kappa0,a1,a2,alpha1,alpha2,PI,kappa
509   REAL*8, DIMENSION(3,3) :: IDE,SIG,DAM,DfY,DfS
510   REAL*8, DIMENSION(6,6) :: CM
511   REAL*8, DIMENSION(9,9) :: DSISI,DSID,ATS
512   REAL*8, DIMENSION(9)   :: VDfY,VDfS
513   !****************************************
514   ! Get derivatives needed for CM_ATS
515   CALL DERY(E,nu,A,B,k1,k2,coms,alpha1,alpha2,PI,IDE,SIG,DfY)
516   CALL DERSIG(E,nu,A,B,k1,k2,coms,alpha1,alpha2,PI,IDE,SIG,DfS)
517   CALL VEC(DfY,VDfY)
518   CALL VEC(DfS,VDfS)
519   CALL SIGDAM(E,nu,alpha1,alpha2,SIG,DSID)
520   CALL SIGSIG(E,nu,alpha1,alpha2,DAM,DSISI)
521
522   ! Cross product DfY X DfY
523   DO II = 1,9
524     DO JJ = 1,9
525       ATS(II,JJ) = VDfY(II)*VDfY(JJ)
526     END DO
527   END DO
528
529   ! Multiply it with DSID from the DSID^T from the right side
530   DSID = TRANSPOSE(DSID)
531   ATS  = MATMUL(ATS,DSID)
532
533   ! Add cross product DfY X DfS
534   DO II = 1,9
535     DO JJ = 1,9
536       ATS(II,JJ) = ATS(II,JJ) + VDfY(II)*VDfS(JJ)
537     END DO
538   END DO
539
540   ! Multiply with DSID
541   ATS = MATMUL(DSID,ATS)
542
543   ! Multiply with 1/H
544   ATS = ATS*((a2*kappa**2+kappa0**2)**2 &
545         /(H0*kappa0*(-a2*kappa**2+2*a1*kappa*kappa0+kappa0**2)))
546
547   ! Add DSISI
548   ATS = ATS+DSISI
549
550   ! Reorder indexing and compress to 6*6 matrix form
551   CM(1,1) = ATS(1,1)
552   CM(1,2) = ATS(1,5)
553   CM(1,3) = ATS(1,9)
554   CM(1,4) = ATS(1,2)+ATS(1,4)
```

```
555 CM(1,5) = ATS(1,3)+ATS(1,7)
556 CM(1,6) = ATS(1,6)+ATS(1,8)
557 CM(2,1) = ATS(5,1)
558 CM(2,2) = ATS(5,5)
559 CM(2,3) = ATS(5,9)
560 CM(2,4) = ATS(5,2)+ATS(5,4)
561 CM(2,5) = ATS(5,3)+ATS(5,7)
562 CM(2,6) = ATS(5,6)+ATS(5,8)
563 CM(3,1) = ATS(9,1)
564 CM(3,2) = ATS(9,5)
565 CM(3,3) = ATS(9,9)
566 CM(3,4) = ATS(9,2)+ATS(9,4)
567 CM(3,5) = ATS(9,3)+ATS(9,7)
568 CM(3,6) = ATS(9,6)+ATS(9,8)
569 CM(4,1) = ATS(2,1)
570 CM(4,2) = ATS(2,5)
571 CM(4,3) = ATS(2,9)
572 CM(4,4) = ATS(2,2)+ATS(2,4)
573 CM(4,5) = ATS(2,3)+ATS(2,7)
574 CM(4,6) = ATS(2,6)+ATS(2,8)
575 CM(5,1) = ATS(3,1)
576 CM(5,2) = ATS(3,5)
577 CM(5,3) = ATS(3,9)
578 CM(5,4) = ATS(3,2)+ATS(3,4)
579 CM(5,5) = ATS(3,3)+ATS(3,7)
580 CM(5,6) = ATS(3,6)+ATS(3,8)
581 CM(6,1) = ATS(6,1)
582 CM(6,2) = ATS(6,5)
583 CM(6,3) = ATS(6,9)
584 CM(6,4) = ATS(6,2)+ATS(6,4)
585 CM(6,5) = ATS(6,3)+ATS(6,7)
586 CM(6,6) = ATS(6,6)+ATS(6,8)
587 END SUBROUTINE CM_ATS
588 !**********************************************************************
589
590 !**********************************************************************
591 SUBROUTINE DERY(E,nu,A,B,k1,k2,coms,alpha1,alpha2,PI,IDE,SIG,DfY)
592 ! Differentiates the damage surface with respect to Y
593 IMPLICIT NONE
594 REAL*8 :: E,nu,A,B,k1,k2,coms,alpha1,alpha2,PI,TrSIG,TrSIG2,TrY,TrSIGY,&
595   J2,J3,COTET,GAM,D2GAM,D3GAM
596 REAL*8, DIMENSION(3,3) :: IDE,SIG,SIG2,Y,SIGY,DJ2Y,DJ3Y,DfY
597 !**************************************
598 TrSIG  = SIG(1,1)+SIG(2,2)+SIG(3,3)
599 SIG2   = MATMUL(SIG,SIG)
600 Y      = -alpha1*nu*(TrSIG**2)*IDE/(2._8*E)+alpha2*SIG2/E
601 TrSIG2 = SIG2(1,1)+SIG2(2,2)+SIG2(3,3)
602 TrY    = Y(1,1)+Y(2,2)+Y(3,3)
603 SIGY   = MATMUL(SIG,Y)
604 TrSIGY = SIGY(1,1)+SIGY(2,2)+SIGY(3,3)
605 J2     = 0.5_8*(E*TrY/alpha2+((1.5_8*alpha1*nu/alpha2)-(1._8/3._8)) &
```

```
606           *(TrSIG**2))
607   J3      = ((E/alpha2)*(TrSIGY-TrSIG*TrY)+((2._8/9._8) &
608           -(alpha1*nu/alpha2))*(TrSIG)**3)/3._8
609   DJ2Y    = 0.5_8*E*IDE/alpha2
610   DJ3Y    = E*(SIG-TrSIG*IDE)/(3._8*alpha2)
611   COTET   = 1.5_8*DSQRT(3._8)*J3*(J2)**(-1.5_8)
612
613   IF (COTET > 0._8) THEN
614     GAM   = k1*DCOS((1._8/3._8)*DACOS(k2*COTET))
615     D2GAM = -k1*k2*(3._8*DSQRT(3._8)/4._8)*DSIN((1._8/3._8) &
616             *DACOS(k2*COTET))*J3*(J2**(-2.5_8))/DSQRT(1._8-(k2*COTET)**2)
617     D3GAM = k1*k2*0.5_8*DSQRT(3._8)*DSIN((1._8/3._8) &
618             *DACOS(k2*COTET))*(J2**(-1.5_8))/DSQRT(1._8-(k2*COTET)**2)
619   ELSE
620     GAM   = k1*DCOS(PI/3._8-(1._8/3._8)*DACOS(-k2*COTET))
621     D2GAM = -k1*k2*(3._8*DSQRT(3._8)/4._8) &
622             *DSIN(PI/3._8-(1._8/3._8)*DACOS(-k2*COTET))*J3*(J2**(-2.5_8))&
623             /DSQRT(1._8-(-k2*COTET)**2)
624     D3GAM = k1*k2*0.5_8*DSQRT(3._8)*DSIN((PI/3._8)-(1._8/3._8) &
625             *DACOS(-k2*COTET))*(J2**(-1.5_8))/DSQRT(1._8-(-k2*COTET)**2)
626   END IF
627
628   DfY = DJ2Y*A/coms+DSQRT(J2)*(D2GAM*DJ2Y+D3GAM*DJ3Y) &
629         +GAM*0.5_8*DJ2Y*J2**(-0.5_8)
630   END SUBROUTINE DERY
631   !*********************************************************************
632
633   !*********************************************************************
634   SUBROUTINE DERSIG(E,nu,A,B,k1,k2,coms,alpha1,alpha2,PI,IDE,SIG,DfS)
635   ! Differentiates the damage surface with respect to Sigma
636   IMPLICIT NONE
637   REAL*8 :: E,nu,A,B,k1,k2,coms,alpha1,alpha2,PI,TrSIG,TrSIG2,TrY,TrSIGY,&
638     J2,J3,COTET,GAM,D2GAM,D3GAM
639   REAL*8, DIMENSION(3,3) :: IDE,SIG,SIG2,Y,SIGY,DJ2S,DJ3S,DfS
640   !***************************************
641   TrSIG   = SIG(1,1)+SIG(2,2)+SIG(3,3)
642   SIG2    = MATMUL(SIG,SIG)
643   Y       = -alpha1*nu*(TrSIG**2)*IDE/(2._8*E)+alpha2*SIG2/E
644   TrSIG2  = SIG2(1,1)+SIG2(2,2)+SIG2(3,3)
645   TrY     = Y(1,1)+Y(2,2)+Y(3,3)
646   SIGY    = MATMUL(SIG,Y)
647   TrSIGY  = SIGY(1,1)+SIGY(2,2)+SIGY(3,3)
648   J2      = 0.5_8*(E*TrY/alpha2+((1.5_8*alpha1*nu/alpha2)-(1._8/3._8)) &
649             *(TrSIG**2))
650   J3      = ((E/alpha2)*(TrSIGY-TrSIG*TrY)+((2._8/9._8) &
651             -(alpha1*nu/alpha2))*(TrSIG)**3)/3._8
652   DJ2S    = ((1.5_8*alpha1*nu/alpha2)-(1._8/3._8))*TrSIG*IDE
653   DJ3S    = (E/(3*alpha2))*(Y-TrY*IDE)+IDE*(2._8/9._8-alpha1*nu/alpha2) &
654             *(TrSIG)**2
655   COTET   = 1.5_8*DSQRT(3._8)*J3*(J2)**(-1.5_8)
656
```

```
657 IF (COTET > 0._8) THEN
658   GAM   = k1*DCOS((1._8/3._8)*DACOS(k2*COTET))
659   D2GAM = -k1*k2*(3._8*DSQRT(3._8)/4._8)*DSIN((1._8/3._8) &
660           *DACOS(k2*COTET))*J3*(J2**(-2.5_8))/DSQRT(1._8-(k2*COTET)**2)
661   D3GAM = k1*k2*0.5_8*DSQRT(3._8)*DSIN((1._8/3._8) &
662           *DACOS(k2*COTET))*(J2**(-1.5_8))/DSQRT(1._8-(k2*COTET)**2)
663 ELSE
664   GAM   = k1*DCOS(PI/3._8-(1._8/3._8)*DACOS(-k2*COTET))
665   D2GAM = -k1*k2*(3._8*DSQRT(3._8)/4._8) &
666           *DSIN(PI/3._8-(1._8/3._8)*DACOS(-k2*COTET))*J3*(J2**(-2.5_8))&
667           /DSQRT(1._8-(-k2*COTET)**2)
668   D3GAM = k1*k2*0.5_8*DSQRT(3._8)*DSIN((PI/3._8)-(1._8/3._8) &
669           *DACOS(-k2*COTET))*(J2**(-1.5_8))/DSQRT(1._8-(-k2*COTET)**2)
670 END IF
671
672 DfS = DJ2S*A/coms+DSQRT(J2)*(D2GAM*DJ2S+D3GAM*DJ3S) &
673       +GAM*0.5_8*DJ2S*J2**(-0.5_8)+B*IDE
674 END SUBROUTINE DERSIG
675 !***********************************************************************
676
677 !***********************************************************************
678 SUBROUTINE SIGDAM(E,nu,alpha1,alpha2,SIG,DSID)
679 ! Differentiates free energy with respect to DAM and SIG
680 IMPLICIT NONE
681 REAL(8)  E,nu,alpha1,alpha2,TrSIG
682 REAL(8), DIMENSION(3,3) :: SIG
683 REAL(8), DIMENSION(9,9) :: DSID1,DSID2,DSID
684 !************************************
685 DSID1 = 0._8
686 DSID1(1,1) = 2._8*SIG(1,1)
687 DSID1(1,2) = SIG(1,2)
688 DSID1(2,1) = DSID1(1,2)
689 DSID1(1,3) = SIG(1,3)
690 DSID1(3,1) = DSID1(1,3)
691 DSID1(1,4) = SIG(1,2)
692 DSID1(4,1) = DSID1(1,4)
693 DSID1(1,7) = SIG(1,3)
694 DSID1(7,1) = DSID1(1,7)
695 DSID1(2,2) = SIG(1,1)+SIG(2,2)
696 DSID1(2,3) = SIG(3,2)
697 DSID1(3,2) = DSID1(2,3)
698 DSID1(2,5) = SIG(1,2)
699 DSID1(5,2) = DSID1(2,5)
700 DSID1(2,8) = SIG(1,3)
701 DSID1(8,2) = DSID1(2,8)
702 DSID1(3,3) = SIG(1,1)+SIG(3,3)
703 DSID1(3,6) = SIG(1,2)
704 DSID1(6,3) = DSID1(3,6)
705 DSID1(3,9) = SIG(1,3)
706 DSID1(9,3) = DSID1(3,9)
707 DSID1(4,4) = SIG(1,1)+SIG(2,2)
```

```
DSID1(4,5) = SIG(2,1)
DSID1(5,4) = DSID1(4,5)
DSID1(4,6) = SIG(1,3)
DSID1(6,4) = DSID1(4,6)
DSID1(4,7) = SIG(2,3)
DSID1(7,4) = DSID1(4,7)
DSID1(5,5) = 2._8*SIG(2,2)
DSID1(5,6) = SIG(3,2)
DSID1(6,5) = DSID1(5,6)
DSID1(5,8) = SIG(2,3)
DSID1(8,5) = DSID1(5,8)
DSID1(6,6) = SIG(2,2)+SIG(3,3)
DSID1(6,9) = SIG(2,3)
DSID1(9,6) = DSID1(6,9)
DSID1(7,7) = SIG(1,1)+SIG(3,3)
DSID1(7,8) = SIG(2,1)
DSID1(8,7) = DSID1(7,8)
DSID1(7,9) = SIG(1,3)
DSID1(9,7) = DSID1(7,9)
DSID1(8,8) = SIG(2,2)+SIG(3,3)
DSID1(8,9) = SIG(3,2)
DSID1(9,8) = DSID1(8,9)
DSID1(9,9) = 2._8*SIG(3,3)
DSID1 = (alpha2/E)*DSID1

TrSIG = SIG(1,1)+SIG(2,2)+SIG(3,3)
DSID2 = 0._8
DSID2(1,1) = 1._8
DSID2(1,5) = 1._8
DSID2(1,9) = 1._8
DSID2(5,1) = 1._8
DSID2(5,5) = 1._8
DSID2(5,9) = 1._8
DSID2(9,1) = 1._8
DSID2(9,5) = 1._8
DSID2(9,9) = 1._8
DSID2 = (-alpha1*nu/E*TrSIG)*DSID2
DSID  = DSID1+DSID2
END SUBROUTINE SIGDAM
!***********************************************************************

!***********************************************************************
SUBROUTINE SIGSIG(E,nu,alpha1,alpha2,DAM,DSISI)
! Differentiates free energy twice with respect to Sigma
IMPLICIT NONE
INTEGER II
REAL(8)  E,nu,alpha1,alpha2,TrDAM
REAL(8), DIMENSION(3,3) :: DAM
REAL(8), DIMENSION(9,9) :: DSISI1,DSISI2,DSISI3,DSISI
!***********************************
DSISI1 = 0._8
```

```
759  DSISI1(1,1) = 1._8
760  DSISI1(1,5) = 1._8
761  DSISI1(1,9) = 1._8
762  DSISI1(5,1) = 1._8
763  DSISI1(5,5) = 1._8
764  DSISI1(5,9) = 1._8
765  DSISI1(9,1) = 1._8
766  DSISI1(9,5) = 1._8
767  DSISI1(9,9) = 1._8
768  TrDAM  = DAM(1,1)+DAM(2,2)+DAM(3,3)
769  DSISI1 = (-nu/E*(1._8+alpha1*TrDAM))*DSISI1
770
771  DSISI2 = 0._8
772  DO II = 1,9
773    DSISI2(II,II) = 1._8
774  END DO
775  DSISI2 = ((1._8+nu)/E)*DSISI2
776
777  DSISI3 = 0._8
778  DSISI3(1,1) = 2*DAM(1,1)
779  DSISI3(1,2) = DAM(2,1)
780  DSISI3(2,1) = DSISI3(1,2)
781  DSISI3(1,3) = DAM(3,1)
782  DSISI3(3,1) = DSISI3(1,3)
783  DSISI3(1,4) = DAM(1,2)
784  DSISI3(4,1) = DSISI3(1,4)
785  DSISI3(1,7) = DAM(1,3)
786  DSISI3(7,1) = DSISI3(1,7)
787  DSISI3(2,2) = DAM(1,1)+DAM(2,2)
788  DSISI3(2,3) = DAM(3,2)
789  DSISI3(3,2) = DSISI3(2,3)
790  DSISI3(2,5) = DAM(1,2)
791  DSISI3(5,2) = DSISI3(2,5)
792  DSISI3(2,8) = DAM(1,3)
793  DSISI3(8,2) = DSISI3(2,8)
794  DSISI3(3,3) = DAM(1,1)+DAM(3,3)
795  DSISI3(3,6) = DAM(1,2)
796  DSISI3(6,3) = DSISI3(3,6)
797  DSISI3(3,9) = DAM(1,3)
798  DSISI3(9,3) = DSISI3(3,9)
799  DSISI3(4,4) = DAM(2,2)+DAM(1,1)
800  DSISI3(4,5) = DAM(2,1)
801  DSISI3(5,4) = DSISI3(4,5)
802  DSISI3(4,6) = DAM(3,1)
803  DSISI3(6,4) = DSISI3(4,6)
804  DSISI3(4,7) = DAM(2,3)
805  DSISI3(7,4) = DSISI3(4,7)
806  DSISI3(5,5) = 2._8*DAM(2,2)
807  DSISI3(5,6) = DAM(3,2)
808  DSISI3(6,5) = DSISI3(5,6)
809  DSISI3(5,8) = DAM(2,3)
```

```
DSISI3(8,5) = DSISI3(5,8)
DSISI3(6,6) = DAM(2,2)+DAM(3,3)
DSISI3(6,9) = DAM(2,3)
DSISI3(9,6) = DSISI3(6,9)
DSISI3(7,7) = DAM(3,3)+DAM(1,1)
DSISI3(7,8) = DAM(2,1)
DSISI3(8,7) = DSISI3(7,8)
DSISI3(7,9) = DAM(3,1)
DSISI3(9,7) = DSISI3(7,9)
DSISI3(8,8) = DAM(3,3)+DAM(2,2)
DSISI3(8,9) = DAM(3,2)
DSISI3(9,8) = DSISI3(8,9)
DSISI3(9,9) = 2._8*DAM(3,3)
DSISI3 = (alpha2/E)*DSISI3
DSISI  = DSISI1+DSISI2+DSISI3
END SUBROUTINE SIGSIG
!***********************************************************************

!***********************************************************************
SUBROUTINE MAT(VEC1,MAT1)
! Takes a vector and writes it in a symmetric matrix form
IMPLICIT NONE
REAL(8), DIMENSION(3,3) :: MAT1
REAL(8), DIMENSION(9)   :: VEC1
!****************************************
MAT1(1,1) = VEC1(1)
MAT1(2,1) = VEC1(2)
MAT1(3,1) = VEC1(3)
MAT1(1,2) = VEC1(4)
MAT1(2,2) = VEC1(5)
MAT1(3,2) = VEC1(6)
MAT1(1,3) = VEC1(7)
MAT1(2,3) = VEC1(8)
MAT1(3,3) = VEC1(9)
END SUBROUTINE MAT
!***********************************************************************

!***********************************************************************
SUBROUTINE VEC(MAT1,VEC1)
! Takes a symmetric matrix and writes in a vector form
IMPLICIT NONE
REAL(8), DIMENSION(3,3)   :: MAT1
REAL(8), DIMENSION(9)     :: VEC1
!****************************************
VEC1(1) = MAT1(1,1)
VEC1(2) = MAT1(2,1)
VEC1(3) = MAT1(3,1)
VEC1(4) = MAT1(1,2)
VEC1(5) = MAT1(2,2)
VEC1(6) = MAT1(3,2)
VEC1(7) = MAT1(1,3)
```

```
VEC1(8) = MAT1(2,3)
VEC1(9) = MAT1(3,3)
END SUBROUTINE VEC
!***********************************************************************

!***********************************************************************
      SUBROUTINE PFACT(W,N,D,IPIV,IFLAG,FSMAL)
!=======================================================================
!     Program to factorize the matrix W(N,N) using Gaussian
!     elimination with scaled partial pivoting
!-----------------------------------------------------------------------
!-----------------------------------------------------------------------
!    Reference:
!     S.D. Conte and C. de Boor, Elementary numerical analysis -
!     An algorithmic approach, McGraw-Hill, 1987, Chapter 4.4
!=======================================================================
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION W(N,*),D(*),IPIV(*)
      AMAX1(X,Y)=DMAX1(X,Y)
      ABS(X)=DABS(X)
      IFLAG = 1
      DO 1009 I=1,N
         IPIV(I)=I
         ROWMAX=0.
         DO 1005 J=1,N
            ROWMAX=AMAX1(ROWMAX,ABS(W(I,J)))
 1005    CONTINUE
         IF(ABS(ROWMAX).LT.FSMAL) THEN
            IFLAG=-2*N
            ROWMAX=1.
         END IF
         D(I)=ROWMAX
 1009 CONTINUE
      IF(N.LE.1) RETURN
      DO 1020 K=1,N-1
         COLMAX=ABS(W(K,K))/D(K)
         ISTAR=K
         DO 1013 I=K+1,N
            AWIKOD=ABS(W(I,K))/D(I)
            IF(AWIKOD.GT.COLMAX) THEN
               COLMAX=AWIKOD
               ISTAR=I
            END IF
 1013    CONTINUE
         IF(ABS(COLMAX).LE.FSMAL) THEN
            IFLAG=-K
         ELSE
            IF(ISTAR.GT.K) THEN
               I=IPIV(ISTAR)
               IPIV(ISTAR)=IPIV(K)
               IPIV(K)=I
```

```
912                TEMP=D(ISTAR)
913                D(ISTAR)=D(K)
914                D(K)=TEMP
915                DO 1015 J=1,N
916                   TEMP=W(ISTAR,J)
917                   W(ISTAR,J)=W(K,J)
918                   W(K,J)=TEMP
919  1015          CONTINUE
920             END IF
921             DO 1019 I=K+1,N
922                W(I,K)=W(I,K)/W(K,K)
923                RATIO=W(I,K)
924                DO 1018 J=K+1,N
925                   W(I,J)=W(I,J)-RATIO*W(K,J)
926  1018          CONTINUE
927  1019       CONTINUE
928          END IF
929  1020 CONTINUE
930       IF(ABS(W(N,N)).LT.FSMAL) IFLAG=-N
931       RETURN
932       END SUBROUTINE PFACT
933  !*********************************************************************
934       SUBROUTINE SUBST(W,X,B,IPIV,N)
935  !=====================================================================
936  !    Program to get the solution of WX=B by backsubstitution
937  !=====================================================================
938       IMPLICIT REAL*8 (A-H,O-Z)
939       DIMENSION W(N,*),X(*),B(*),IPIV(*)
940       IF(N.LE.1) THEN
941          X(1)=B(1)/W(1,1)
942          RETURN
943       END IF
944       IP=IPIV(1)
945       X(1)=B(IP)
946       DO 1015 I=2,N
947          SUM=0.D0
948          DO 1014 J=1,I-1
949             SUM=SUM+W(I,J)*X(J)
950  1014     CONTINUE
951          IP=IPIV(I)
952          X(I)=B(IP)-SUM
953  1015 CONTINUE
954       X(N)=X(N)/W(N,N)
955       DO 1020 I=N-1,1,-1
956          SUM=0.D0
957          DO 1019 J=I+1,N
958             SUM=SUM+W(I,J)*X(J)
959  1019     CONTINUE
960          X(I)=(X(I)-SUM)/W(I,I)
961  1020 CONTINUE
962       RETURN
```

```
963         END SUBROUTINE SUBST
964   !*****************************************************************
965         SUBROUTINE INVERT(A,AI,IPIV,N,X,FSMAL,IOUT)
966   !=================================================================
967   !    Program to invert rectangular matrix A
968   !=================================================================
969         IMPLICIT REAL*8 (A-H,O-Z)
970         DIMENSION A(N,*),AI(N,*),X(*),IPIV(*)
971         CALL PFACT(A,N,X,IPIV,IFLAG,FSMAL)
972         IF(IFLAG.EQ.0) THEN
973            WRITE(IOUT,5000)
974            STOP
975         END IF
976         DO 1100 I=1,N
977            X(I)=0.D0
978    1100 CONTINUE
979         DO 1200 J=1,N
980            X(J)=1.D0
981            CALL SUBST(A,AI(1,J),X,IPIV,N)
982            X(J)=0.
983    1200 CONTINUE
984         RETURN
985    5000 FORMAT('UMAT: NUMERICAL ERROR - TRYING TO INVERT SINGULAR MATRIX')
986         END SUBROUTINE INVERT
987   !*****************************************************************
988
989   !====================================
990   ! E:          Elastic modulus
991   ! nu:         Poisson's ratio
992   ! A:          Material parameter A
993   ! B:          Material parameter B
994   ! k1:         Material parameter k_1
995   ! k2:         Material parameter k_2
996   ! coms:       Material parameter \sigMa_{c0}
997   ! H0:         Material parameter H_0
998   ! kappa0:     Material parameter \kappa_0
999   ! a1:         Material parameter a_1
1000  ! a2:         Material parameter a_2
1001  ! alpha1:     Material parameter \alpha_1
1002  ! alpha2:     Material parameter \alpha_2
1003  ! MaxNRiters: Max n:o of iterations of the consistency condition
1004
1005  !====================================
1006  ! SIG:    Stress tensor
1007  ! SIGO:   Initial Stress tensor at the beginning of increment
1008  ! SIGVEC: SIG in vector form
1009  ! EPS:    Current Strain tensor
1010  ! EPSO:   Initial Strain tensor at the beginning of increment
1011  ! DEPS:   Strain tensor increment
1012  ! EPSVEC: EPS in vecrot form
1013  ! Y:      Thermodynamic force Y
```

```
1014 ! DAM:    Damage tensor
1015 ! kappa:  Internal variable \kappa
1016 ! J2:     Deviatoric stress invariant J2
1017 ! J3:     Deviatoric stress invariant J3
1018 ! GAM:    \Lambda in the thesis
1019 ! COTET:  Lode angle
1020 ! f:      Damage surface
1021
1022 !====================================
1023 ! IDE:  Identity matrix
1024 ! SIG2: SIG^2
1025 ! SIG3: SIG^3
1026 ! SIGY: SIG*Y
1027 ! SIGDEV: Deviatoric stress tensor s
1028 ! SIGDE2: SIGDEV^2
1029 ! SIGDE3: SIGDEV^3
1030
1031 !====================================
1032 ! TrSIG:  trace of SIG
1033 ! TrSIG2: trace of SIG^2
1034 ! TrEPS:  trace of EPS
1035 ! TrEPSD: trace of EPSD
1036 ! TrDAM:  trace of DAM
1037 ! TrY:    trace of Y
1038 ! TrSIGY: trace of SIG*Y
1039
1040 !====================================
1041 ! CM:   Compliance matrix or algorithmic tangent stiffness matrix
1042 ! eta1: Term \eta_1 in compliance matrix
1043 ! eta2: Term \eta_2 in compliance matrix
1044 ! eta3: Term \eta_3 in compliance matrix
1045
1046 !====================================
1047 ! JAC:   Jacobian for Newton Raphson iteration
1048 ! D2GAM: Derivative of GAM with respect to J2
1049 ! D3GAM: Derivative of GAM with respect to J3
1050 ! DJ2Y:   Derivative of J2 with respect to Y
1051 ! DJ3Y:   Derivative of J3 with respect to Y
1052 ! DJ2S:  Derivative of J2 with respect to SIG
1053 ! DJ3S:  Derivative of J3 with respect to SIG
1054 ! DfY:   Derivative of the damage surface with respect to Y
1055 ! DfS:   Derivative of the damage surface with respect to SIG
1056 ! DSID:  Derivative of the free energy with respect to SIG and DAM
1057 ! DSISI: Derivative of the free energy twice with respect to SIG
1058
1059 !====================================
1060 ! Dlam:    Multiplier \dot\lambda in the thesis
1061 ! DDlam:   Iterative change of \lambda
1062 ! DDkappa: Iterative change of \kappa
1063 ! DDAM:    Iterative change of DAM
```